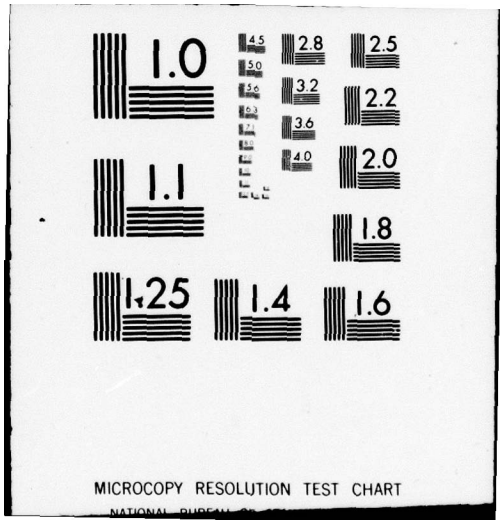
AD-A080 238 AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH SCHOO--ETC F/G 17/5
AN ADAPTIVE DISTRIBUTED-MEASUREMENT EXTENDED KALMAN FILTER FOR --ETC(U)
DEC 79 R L JENSEN, D A HARNLY
UNCLASSIFIED AFIT/GA/EE/79-1-VOL-2 NL

1 OF 3
AD A
080238


1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0
4.5
5.0
5.6
6.3
7.1
8.0
9.0
MICROCOPY RESOLUTION TEST CHART

LEVEL II

UNITED STATES AIR FORCE

AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

DISCLAIMER NOTICE

THIS DOCUMENT IS BEST QUALITY PRACTICABLE. THE COPY FURNISHED TO DDC CONTAINED A SIGNIFICANT NUMBER OF PAGES WHICH DO NOT REPRODUCE LEGIBLY.

(14) AFIT/GA/EE/79-1-Vol-2

(9) Master's Thesis

(6) AN ADAPTIVE DISTRIBUTED-MEASUREMENT EXTENDED KALMAN FILTER FOR A SHORT RANGE TRACKER. Volume II.

THESIS

AFIT/GA/EE/79-1

Vol II

(10) Robert L. Jensen
Capt USAF

Douglas A. Harnly
Capt USAF

(11) Dec 79

(12) 263

Approved for public release; distribution unlimited

AFIT/GA/EE/79-1

AN ADAPTIVE DISTRIBUTED-MEASUREMENT EXTENDED KALMAN FILTER FOR A SHORT RANGE TRACKER

VOLUME II

THESIS

Presented to the Faculty of the School of Engineering
of the Air Force Institute of Technology
Air University
in Partial Fulfillment of the
Requirements for the Degree of
Master of Science

by

Robert L. Jensen
Capt USAF

and

Douglas A. Harnly
Capt USAF

Graduate Astronautical Engineering
December 1979

Approved for public release; distribution unlimited.

Volume II

Contents

		Page
List of Figures		iii
List of Tables		v
List of Symbols		vi
Appendix A:	Previous Truth Model and Filter Equations	1
Appendix B:	Real Data Analysis for Spatial Correlations of Background Noise	11
Appendix C:	Real Data Analysis for Temporal Correlations of Background Noise	17
Appendix D:	$Q_{(Fd)}$ Calculation	21
Appendix E:	Computer Simulation Considerations	26
Appendix F:	FORTRAN Code: Main Programs and Library of Subroutines	31
Appendix G:	FORTRAN Code: Plot Programs	71
Appendix H:	Input Parameters for Cases 1-31	79
Appendix I:	Performance Plots for Cases 1-12	85
Appendix J:	Performance Plots for Cases 13-31	144
Appendix K:	Input Parameters for Cases 32-37	215
Appendix L:	Performance Plots for Cases 32-37	217

Volume II

List of Figures

Figure		Page
E-1	Convergence of Target Position Standard Deviation	29
1	Case 1 Performance Plots	86
2	Case 2 Performance Plots	92
3	Case 3 Performance Plots	104
4	Case 4 Performance Plots	108
5	Case 5 Performance Plots	112
6	Case 6 Performance Plots	116
7	Case 7 Performance Plots	120
8	Case 8 Performance Plots	124
9	Case 9 Performance Plots	128
10	Case 10 Performance Plots	132
11	Case 11 Performance Plots	136
12	Case 12 Performance Plots	140
13	Case 13 Performance Plots	145
14	Case 14 Performance Plots	149
15	Case 15 Performance Plots	153
16	Case 16 Performance Plots	157
17	Case 17 Performance Plots	161
18	Case 18 Performance Plots	165
19	Case 19 Performance Plots	169
20	Case 20 Performance Plots	173
21	Case 21 Performance Plots	177
22	Case 23 Performance Plots	183

Volume II

List of Figures

Figure		Page
23	Case 25 Performance Plots	187
24	Case 26 Performance Plots	191
25	Case 27 Performance Plots	195
26	Case 28 Performance Plots	199
27	Case 29 Performance Plots	203
28	Case 30 Performance Plots	207
29	Case 31 Performance Plots	211
30	Case 32 Performance Plots	218
31	Case 33 Performance Plots	222
32	Case 34 Performance Plots	228
33	Case 35 Performance Plots	234
34	Case 36 Performance Plots	240
35	Case 37 Performance Plots	248

Volume II

List of Tables

Table		Page
B-I	First Horizontal Neighbor Correlations . . .	13
B-II	Second Horizontal Neighbor Correlations . .	14
B-III	First Vertical Neighbor Correlations: A to B	15
B-IV	Second Vertical Neighbor Correlations: A to A and B to B	16
C-I	Temporal Correlation Coefficient Data . . .	20

List of Symbols

$\underline{A}$	General parameter vector
A_p	Area of picture element
AR	Aspect ratio: $AR = \sigma_v/\sigma_{pv}$
$\underline{b}$	Realization of parameter vector
$\underline{B}$	System control input matrix
$E[\cdot]$	Expected value
$\underline{F}$	System plant matrix
$\underline{G}$	System noise input matrix
$\underline{H}$	Linearization of intensity measurements
$\underline{h}(\ )$	Nonlinear measurement relation
Hz	Hertz
I_F	Filter maximum intensity
I_{max}	Maximum target intensity
$\underline{K}$	Kalman filter gain matrix
L	Likelihood function
L'	Full scale maximum likelihood equation
$\underline{P}$	Covariance matrix
$\underline{P}_L$	Covariance of likelihood function
$\underline{Q}$	Strength of disturbance process matrix
q	Noise strength
$\underline{R}$	Covariance of measurement noise matrix
r	Correlation coefficient
$\underline{r}$	Residual vector
R_F	Measurement noise variance
S/N	Signal to noise ratio: $S/N = I_{max}/\sigma_N$

List of Symbols

t	Time
$\underline{u}$	Deterministic velocity input function
$\underline{v}$	General velocity vector
w	White noise
$\hat{\underline{x}}$	Filter state estimate vector
$\underline{x}$	State vector
$\hat{x}_A$	Estimate of horizontal position due to atmospheric jitter
$\hat{x}_D$	Estimate of horizontal velocity due to target motion
$\hat{\dot{x}}_D$	Estimate of horizontal velocity due to target motion
x_{peak}, x_p	Horizontal coordinate of Gaussian intensity function maximum
$\hat{y}_A$	Estimate of vertical position due to atmospheric jitter
$\hat{y}_D$	Estimate of vertical position due to target motion
$\hat{\dot{y}}_D$	Estimate of vertical velocity due to target motion
y_{peak}, y_p	Vertical coordinate of Gaussian intensity function maximum
$\underline{z}$	General measurement vector
α	Azimuth
β	Elevation
Γ	Vector of actual measurements
γ	Angle between velocity vector and image plane
$\Delta\underline{x}$	Filter state vector update
Δx_v	Image plane velocity (v_{LOS}) direction coordinate of intensity pattern

List of Symbols

Δy_v	Coordinate perpendicular to image plane velocity (v_{LOS}) of intensity pattern
δ	Intensity peak separation due to acceleration
θ	Orientation angle in image plane
ρ	Line of sight range
σ_A	RMS value of atmospheric jitter (pixels)
σ_F	RMS value of FLIR noise
σ_g	Dispersion of Gaussian intensity function
σ_D	RMS value of target motion (pixels)
σ_N	RMS value of background noise
σ_R	Variance of measurement noise
τ_A	Correlation time of atmospheric jitter
τ_D	Correlation time of target dynamics
τ_N	Correlation time of background noise
Φ	State transition matrix

Subscripts

A	Atmospheric jitter
D	Target dynamics
d	Discrete form
F	Filter
h	Horizontal
I	Inertial
o	Initial value
peak	Maximum intensity position
pv	Direction perpendicular to image plane velocity

List of Symbols

T	Truth model
v	Image plane velocity frame
α,az	Azimuth direction
β,el	Elevation direction

Superscripts

.	Time derivative
^	Estimate
+	After update
-	Before update

Appendix A

Previous Truth Model and Filter Equations (Ref 5:9-29)

Having assumed x_{peak} and y_{peak} to be the sum of x_D and x_A, and y_D and y_A respectively, the previous study further modelled the dynamic components mathematically as (Ref 5:10):

$$\dot{x}_D(t) = -1/\tau_D \, x_D(t) + w_1(t) \qquad \text{(A-1)}$$

$$\dot{y}_D(t) = -1/\tau_D \, y_D(t) + w_2(t) \qquad \text{(A-2)}$$

where $$E\{w_1(t)\} = E\{w_2(t)\} = 0 \quad \forall t \qquad \text{(A-3)}$$

$$E\{w_1(t)\, w_1(s)\} = E\{w_2(t)\, w_2(s)\} = 2\sigma_d^2/\tau_D \, \delta(t-s) \qquad \text{(A-4)}$$

$$E\{w_1(t)\, w_2(s)\} = 0 \quad \forall t,s \qquad \text{(A-5)}$$

and τ_D = correlation time

$w_1(t), w_2(t)$ = independent, white, Gaussian noise processes

σ_D^2 = desired sigma or root mean square (RMS) value on the outputs x_D and y_D.

The atmospheric jitter components, x_A and y_A were modelled as the outputs of third order shaping filters driven by white, Gaussian noise. This shaping filter is represented in the frequency domain by the following transfer function (Ref 5:10,12):

$$\frac{Y}{W} = \frac{Kab^2}{(s+a)(s+b)^2} \qquad \text{(A-6)}$$

where $a = 14.14$ rad/sec

$b = 659.5$ rad/sec

K = gain adjusted to obtain desired root mean square (RMS) jitter output

and
$$E\{w(t)\} = 0 \tag{A-7}$$

$$E\{w(t)\ w(s)\} = 1\ \delta(t-s) \tag{A-8}$$

Incorporating these models into state space notation yielded a time invariant form (Ref 5:11,13) for the truth model

$$\dot{\underline{x}}_T(t) = \underline{F}_T\underline{x}_T(t) + \underline{G}_T\underline{w}_T(t) \tag{A-9}$$

where $\underline{F}_T$ = 8 by 8 truth model plant matrix

$\underline{x}_T$ = truth model state vector with (8 components)

$\underline{G}_T$ = truth model input matrix (8 by 4)

$\underline{w}_T$ = vector of white Gaussian noise inputs (4 by 1)

and
$$E\{\underline{w}_T(t)\} = 0 \tag{A-10}$$

$$E\{\underline{w}_T(t)\ \underline{w}_T^T(s)\} = \underline{Q}_T\ \delta(t-s) \tag{A-11}$$

$$\underline{Q}_T = \begin{bmatrix} \frac{2\sigma_D^2}{\tau_D} & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \frac{2\sigma_D^2}{\tau_D} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{A-12}$$

Mercier had numerical problems with the standard phase variable form for A-9 but successfully used the Jordon canonical form as follows (Ref 5:13-14):

$$\dot{\underline{x}}_T(t) = \begin{bmatrix} -1/\tau_D & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -a & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -b & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -b & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1/\tau_D & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -a & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -b & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -b \end{bmatrix} \underline{x}_T(t)$$

$$+ \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & G_1 & 0 & 0 \\ 0 & G_2 & 0 & 0 \\ 0 & G_3 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & G_1 \\ 0 & 0 & 0 & G_2 \\ 0 & 0 & 0 & G_3 \end{bmatrix} \underline{w}_T(t) \qquad \text{(A-13)}$$

where $G_1 = Kab^2/(a-b)^2$

$G_2 = -G_1$

$G_3 = Kab^2/(a-b)$

and the output matrix is then described by

$$\underline{y}_T(t) = \begin{bmatrix} x_{peak}(t) \\ y_{peak}(t) \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \end{bmatrix} \underline{x}_T(t) \qquad \text{(A-14)}$$

The state transition matrix corresponding to A-13 is block diagonal due to the independence of the two channels. Each 4 by 4 diagonal block is equal to (Ref 5:14-15)

$$\Phi_{Tx}(t_{i+1},t_i) = \Phi_{Ty}(t_{i+1},t_i)$$

$$= \begin{bmatrix} e^{-\Delta t/\tau_D} & 0 & 0 & 0 \\ 0 & e^{-a\Delta t} & 0 & 0 \\ 0 & 0 & e^{-b\Delta t} & \Delta t\varepsilon^{-b\Delta t} \\ 0 & 0 & 0 & e^{-b\Delta t} \end{bmatrix} \quad \text{(A-15)}$$

So, an equivalent discrete time model of the time propagation of $\underline{x}_T(t)$ is given by (Ref 5:15)

$$\underline{x}_T(t_{i+1}) = \underline{\Phi}_T\underline{x}_T(t_i) + \underline{w}_{Td}(t_i) \quad \text{(A-16)}$$

where $$\underline{w}_{Td}(t_i) = \int_{t_i}^{t_{i+1}} \underline{\Phi}_T(t_{i+1},\tau)\ \underline{G}_T(\tau)\underline{w}_T(\tau)\,d\tau \quad \text{(A-17)}$$

as commonly used but $\underline{w}_{Td}$ is more properly defined

$$\underline{w}_{Td}(t_i) = \int_{t_i}^{t_{i+1}} \underline{\Phi}_T(t_{i+1},\tau)\ \underline{G}_T(\tau)\ \ d\underline{\beta}(\tau) \text{(Ref 8:171)} \quad \text{(A-17a)}$$

$$E\{\underline{w}_{Td}(t_i)\} = 0\ ;\ E\left\{\underline{w}_{Td}(t_i)\underline{w}_{Td}^T(t_j)\right\} = \underline{Q}_{Td}\delta_{ij} \quad \text{(A-18)}$$

$$\underline{Q}_{Td} = \int_{t_i}^{t_{i+1}} \underline{\Phi}_T(t_{i+1},\tau)\underline{G}_T(\tau)\underline{Q}_T(\tau)\underline{G}_T^T(\tau)\underline{\Phi}_T^T(t_{i+1},\tau)\,d\tau \quad \text{(A-19)}$$

For simulation implementation, A-16 was used in the following form (Ref 5:16)

$$\underline{x}_T(t_{i+1}) = \underline{\Phi}_T \underline{x}_T(t_i) + {}^c\sqrt{\underline{Q}_{Td}}\, \underline{w}_N(t_i) \qquad \text{(A-20)}$$

where ${}^c\sqrt{\underline{Q}_{Td}}$ is a Cholesky square root matrix of $\underline{Q}_{Td}$ such that

$${}^c\sqrt{\underline{Q}_{Td}}\ {}^c\sqrt{\underline{Q}_{Td}}^{\,T} = \underline{Q}_{Td} \qquad \text{(A-21)}$$

and $\underline{w}_N(t_i)$ is a vector of independent Gaussian noises whose terms each have unit variance.

The truth model measurement equation expresses mathematically that each pixel output is the sum of target intensity radiation, FLIR generated noise, and background noise as follows (Ref 5:17):

$$z_{jk}(t_i) = \frac{1}{A_p} \iint_{\substack{jkth \\ pixel}} I_{max} \exp\left\{-\frac{1}{2\sigma_g^2}\left[(x-x_{peak}(t_i))^2 + (y-y_{peak}(t_i))^2\right]\right\} dxdy + n_{jk}(t_i) + f_{jk}(t_i) \qquad \text{(A-22)}$$

where $z_{jk}(t_i)$ = output of jkth pixel at time t_i

A_p = area of a pixel

(x,y) = coordinates of any point in the pixel array

I_{max} = maximum target intensity

σ_g = dispersion of target's Gaussian intensity function

$n_{jk}(t_i)$ = background noise for the jkth pixel

$f_{jk}(t_i)$ = FLIR noise term for the jkth pixel

He assumed that $n_{jk}(t_i)$ and $f_{jk}(t_i)$ were independent wide band noises which could be appropriately modelled as zero mean, white Gaussian processes with the following statistics (Ref 5:18):

$$E\{n_{kl}(t_i)\ n_{kl}(t_j)\} = \sigma_B^2\ \delta_{ij} \qquad \text{(A-23)}$$

$$E\{f_{kl}(t_i)\ f_{kl}(t_j)\} = \sigma_f^2\ \delta_{ij} \qquad \text{(A-24)}$$

$$E\{n_{kl}(t_i)n_{mn}(t_j)\} = E\{f_{kl}(t_i)f_{mn}(t_j)\} = 0$$

$$(kl \neq mn) \qquad \text{(A-25)}$$

Captain Mercier used a four state extended Kalman filter - one state for target dynamics representation and one state for atmospheric jitter representation in each of two directions in the FLIR focal plane. Each state was modelled as a stationary, first order, Gauss-Markov process expressed as (Ref 5:19):

$$\dot{x}(t) = -\frac{1}{\tau}\,x(t) + w(t) \qquad \text{(A-26)}$$

with $E\{w(t)\} = 0\ ;\ E\{w(t)w(s)\} = \frac{2\sigma^2}{\tau}\,\delta(t-s)$ (A-27)

where τ = correlation time

$w(t)$ = white noise process of strength q

σ^2 = desired variance of x(t)

The state space form where $\underline{x}_F^T = [x_D \; x_A \; y_D \; y_A]$ is (Ref 5:20):

$$\dot{\underline{x}}_F(t) = \underline{F}_F\underline{x}(t) + \underline{w}_F(t) \qquad \text{(A-28)}$$

where

$$\underline{F}_F = \begin{bmatrix} -1/\tau_D & 0 & 0 & 0 \\ 0 & -1/\tau_A & 0 & 0 \\ 0 & 0 & -1/\tau_D & 0 \\ 0 & 0 & 0 & -1/\tau_A \end{bmatrix} \qquad \text{(A-29)}$$

$\underline{w}_F(t)$ = input white noise vector of dimension 4

such that $$E\{\underline{w}_F(t)\} = 0 \qquad \text{(A-30)}$$

and

$$E\{\underline{w}_F(t)\underline{w}_F(s)\} = \underline{Q}\delta(t-s) = \begin{bmatrix} q_D & 0 & 0 & 0 \\ 0 & q_A & 0 & 0 \\ 0 & 0 & q_D & 0 \\ 0 & 0 & 0 & q_A \end{bmatrix} \delta(t-s) \qquad \text{(A-31)}$$

The first order system approximation in A-28 and A-29 for the atmospheric jitter described in A-6 is justified based on the discrepancy between the two break frequencies and the dominant effect of the low frequency pole.

With conversion to an equivalent discrete time form of equations (A-28) to (A-31), the filter propagation equations are (Ref 5:20-23):

$$\hat{\underline{x}}_F(t_{i+1}^-) = \underline{\Phi}_F(t_{i+1},t_i)\,\hat{\underline{x}}_F(t_i^+) \qquad \text{(A-32)}$$

$$\underline{P}_F(t_{i+1}^-) = \underline{\Phi}_F(t_{i+1},t_i)\underline{P}_F(t_i^+)\underline{\Phi}_F^T(t_{i+1},t_i)$$

$$+\int_{t_i}^{t_{i+1}} \underline{\Phi}_F(t_{i+1},\tau)\underline{Q}_F(\tau)\underline{\Phi}_F^T(t_{i+1},\tau)d\tau \qquad \text{(A-33)}$$

where $\underline{\Phi}_F(t,t_i)$

$$= \begin{bmatrix} e^{-(t-t_i)/\tau_D} & 0 & 0 & 0 \\ 0 & e^{-(t-t_i)/\tau_A} & 0 & 0 \\ 0 & 0 & e^{-(t-t_i)/\tau_D} & 0 \\ 0 & 0 & 0 & e^{-(t-t_i)/\tau_A} \end{bmatrix} \qquad \text{(A-34)}$$

and $\underline{Q}_F$ is a constant matrix whose entries represent the appropriate strengths of the continuous time noise processes in (A-29) to obtain the proper output variances of σ_D^2 and σ_A^2 (Ref 8:179). Therefore

$$\underline{Q}_F = \begin{bmatrix} \frac{2\sigma_D^2}{\tau_D} & 0 & 0 & 0 \\ 0 & \frac{2\sigma_A^2}{\tau_A} & 0 & 0 \\ 0 & 0 & \frac{2\sigma_D^2}{\tau_D} & 0 \\ 0 & 0 & 0 & \frac{2\sigma_A^2}{\tau_A} \end{bmatrix} \qquad \text{(A-35)}$$

and the second term in A-33 becomes

$$\begin{bmatrix} \sigma_D^2(1-e^{-2\Delta t/\tau_D}) & 0 & 0 & 0 \\ 0 & \sigma_A^2(1-e^{-2\Delta t/\tau_A}) & 0 & 0 \\ 0 & 0 & \sigma_D^2(1-e^{-2\Delta t/\tau_D}) & 0 \\ 0 & 0 & 0 & \sigma_A^2(1-e^{-2\Delta t/\tau_A}) \end{bmatrix}.$$

Similar to A-22, the measurement relation is (Ref 5:24)

$$z_{jk}(t_i) = \frac{1}{A_p} \iint_{\substack{ijth \\ pixel}} I_{max} \exp\left\{-\frac{1}{2\sigma_g^2}\left[(x-x_{peak}(t_i))^2 + (y-y_{peak}(t_i))^2\right]\right\} dxdy + n_{jk}(t_i) + f_{jk}(t_i) \quad \text{(A-36)}$$

where the only difference between A-22 and A-36 is that x_{peak} and y_{peak} are functions of $\underline{x}_p(t_i)$. Since an 8 by 8 tracking window is used the measurements are expressed as a 64-dimensional vector and are assumed to be a nonlinear function of the filter states plus noise (Ref 5:25),

$$\underline{z}(t_i) = \underline{h}(\underline{x}(t_i), t_i) + \underline{v}(t_i) \quad \text{(A-37)}$$

where

$$E\{\underline{v}(t_i)\} = 0 \quad \text{(A-38)}$$

$$E\{\underline{v}(t_i)\underline{v}^T(t_j)\} = R_F \underline{I} \delta_{ij} \quad \text{(A-39)}$$

Thus, the usual form of the extended Kalman filter update equations would be (Ref 5:26)

$$\underline{K}(t_i) = \underline{P}(t_i^-)\underline{H}^T(t_i)[H(t_i)\underline{P}(t_i^-)\underline{H}^T(t_i) + \underline{R}(t_i)]^{-1} \quad \text{(A-40)}$$

$$\hat{\underline{x}}(t_i^+) = \hat{\underline{x}}(t_i^-) + \underline{K}(t_i)\{\underline{\Gamma}(t_i) - \underline{h}(\hat{\underline{x}}(t_i^-), t_i)\} \quad \text{(A-41)}$$

$$\underline{P}(t_i^+) = \underline{P}(t_i^-) - \underline{K}(t_i)\underline{H}(t_i)\underline{P}(t_i^-) \quad \text{(A-42)}$$

where

$$\underline{H}(t_i) = \left.\frac{\partial \underline{h}(\underline{x}, t)}{\partial \underline{x}}\right|_{\underline{x} = \hat{\underline{x}}(t_i^-)} \quad \text{(A-43)}$$

and $\underline{\Gamma}(t_i)$ is a 64-dimensional vector of actual (based on the truth model) measurements. To avoid the inversion of

a 64 by 64 matrix as required by A-38, Captain Mercier chose to implement another form of the update equations known as the inverse covariance form (Ref 8:238-241) shown below:

$$\underline{P}(t_i^+) = [\underline{P}^{-1}(t_i^-) + \underline{H}^T(t_i)\underline{R}^{-1}(t_i)\underline{H}(t_i)]^{-1} \quad \text{(A-44)}$$

$$\underline{K}(t_i) = \underline{P}(t_i^+)\underline{H}^T(t_i)\underline{R}^{-1}(t_i) \quad \text{(A-45)}$$

$$\hat{\underline{x}}(t_i^+) = \hat{\underline{x}}(t_i^-) + \underline{K}(t_i)[\underline{\Gamma}(t_i) - \underline{h}(\hat{\underline{x}}(t_i^-), t_i)] \quad \text{(A-46)}$$

This set of equations requires the inversion of two 4 by 4 matrices. Because the pixel noises are assumed to be independent, the $\underline{R}_F$ is diagonal and its inverse simplifies to (Ref 5:27):

$$\underline{R}_F^{-1}(t_i) = \left(\frac{1}{R_F}\right) \underline{I} \quad \text{(A-47)}$$

which can be precomputed once offline.

The partial derivative matrix in equation A-41 is easily computed, using the fact that the integral is a linear operator, and results in a 64 by 4 matrix (Ref 5: 27-28). The particular components of this matrix can be seen explicitly in subroutine MEASF in Appendix F.

Appendix B

Real Data Analysis for Spatial Correlations of Background Noise

Real digital FLIR data has been analyzed to establish the spatial character of the background noise. The data was from a real exercise of tracking an air-to-air missile conducted in the Spring of 1978 by the Air Force Weapons Laboratory (AFWL). The data from AFWL consisted of a matrix of digital data printed every half second of the test. Each frame or matrix of data is called a record with record numbers running from 113 to 190 (38.5 seconds in total duration). As the missile is flying toward the sensor throughout the exercise, the digital values toward the end of the time span represent the sum of 2-by-2 or 4-by-4 arrays of pixels instead of individual pixels to maintain the missile's image in the limited data field of view. The background initially is clear sky with some ground clutter appearing in the latter segment of frames when the missile motion carried it below the horizon.

The measure of correlation between two sets of data, $\underline{X}$ and $\underline{Y}$, of equal dimension, is most commonly expressed by a correlation coefficient, r. The standard equation for the computation of r is (Ref 11:565):

$$r = \frac{n \sum_{i=1}^{n} X_i Y_i - \left(\sum_{i=1}^{n} X_i\right)\left(\sum_{i=1}^{n} Y_i\right)}{\sqrt{\left[n \sum_{i=1}^{n} X_i^2 - \left(\sum_{i=1}^{n} X_i\right)^2\right]\left[n \sum_{i=1}^{n} Y_i^2 - \left(\sum_{i=1}^{n} Y_i\right)^2\right]}} \qquad \text{(B-1)}$$

Using this relation, the values in selected rows and columns of the real digital data which appeared to be unassociated with the real target image in the frame were evaluated for their correlation coefficient. For example, m values in a given row were evaluated for a horizontal, first neighbor

correlation coefficient by letting X_i in B-1 equal the first m-1 values in the row and Y_i equal the corresponding values displaced one pixel to the right (n=m-1).

Tables B-I through B-IV contain the correlation evaluations that were conducted. Table B-I shows evaluations of pixel correlations for pixels side by side (first neighbor) in the horizontal direction. The data shown is record or frame number, the row or column, how many pixels are summed to get the digital data value, and the correlation coefficient (r). Note that the rows are labeled A and B to reflect that the rows of data have a biased average value when compared to the row above or below it, as was first discussed in Chapter I. Table B-II contains correlation coefficients of each digital value with its second horizontal neighbor. Table B-III shows vertical first neighbor correlations evaluated by considering a column of data. When the data values represent a single pixel output as is the case of the first six entries of Table B-III, the first neighbor vertical correlations represent an attempt to compare A values and B values. Table B-IV represents second neighbor, vertical correlations. Its first six entries, (single pixel values) in contrast to Table B-III, represent correlations of A values with A values and B values with B values.

There are several notable trends, some of which were unexpected, in this data. First, horizontal neighbors correlations using one-pixel values averaged .323 with a standard deviation of .119 which can be well matched by an exponential decay model with a correlation distance of one pixel; similar correlations using four and sixteen pixel values averaged .249 (last six entries of Table B-I) with a standard deviation of .224. Second, horizontal neighbors showed, surprisingly, a slight negative average correlation of -.082 for single pixel values with a standard deviation of .113. For four and sixteen pixel values, second neighbor

TABLE B-I

First Horizontal Neighbor Correlations

Record	Row	No. of Entries per Row	No. of Pixels per Entry	Correlation Coef (r)
114	A1	28	1	.27139
114	B1	28	1	.30296
114	A12	28	1	.48088
114	B12	28	1	.35063
119	A1	30	1	.29152
119	B1	30	1	.43761
119	A12	30	1	.08746
119	B12	30	1	.36134
181	A1	24	4	.32512
181	B1	24	4	-.07193
181	B15	24	4	.5637
181	A16	24	4	.1393
185	A1	24	16	.39189
185	A11	24	16	.1452

One pixel average	.323
standard deviation	.119
Four and sixteen pixel average	.249
standard deviation	.224

horizontal correlations were low, also, averaging .123 with a standard deviation of .163. As mentioned before the vertical correlations were confused by the A and B row biased averages. In Table B-III when A and B values were compared, negative correlations averaging -.193 resulted with a standard deviation of .166. However, when A and B values are summed in the four and sixteen pixel digital values, the average correlation coefficient was .355 with a standard deviation of .198. Second neighbor vertical correlations for single pixel values in Table B-IV were very inconsistent averaging .160 (three coefficients of twelve

TABLE B-II

Second Horizontal Neighbor Correlations

Record	Row	No. of Entries per Row	No. of Pixels per Entry	Correlation Coef (r)
114	A1	28	1	-.2266
114	B1	28	1	-.06488
114	A12	28	1	.03753
114	B12	28	1	-.15656
119	A1	30	1	-.117
119	B1	30	1	-.1955
119	A12	30	1	.0948
119	B12	30	1	-.0283
181	A1	24	4	.1475
181	B1	24	4	-.0197
181	B15	24	4	.1634
181	A16	24	4	.0934
185	A1	24	16	.4052
185	A11	24	16	-.0513

One pixel average	-.082
standard deviation	.113
Four and sixteen pixel average	.123
standard deviation	.163

were negative) with a standard deviation of .261. For four and sixteen pixel digital values, the second neighbor vertical average was .335 with a standard deviation of .155.

Physically, the horizontal correlations were expected to be greater than the vertical correlations due to the horizontal mechanical scanning process of the pixels. This could be verified by an experiment where a well defined background is viewed by the FLIR in its normal orientation and then viewed by the FLIR turned 90 degrees. Spatial analysis of data from both angles would be able to sepor-

TABLE B-III

First Vertical Neighbor Correlations
A to B

Record	Column	No. of Entries per Column	No. of Pixels per Entry	Correlation Coef (r)
114	1	24	1	-.51563
114	2	24	1	-.16357
119	1	24	1	-.1850
119	2	24	1	-.0422
119	29	24	1	-.0989
119	30	24	1	-.1511
181	23	31	4	.6059
181	24	31	4	.13625
185	1	22	16	.41517
179	21	28	4	.44834
179	22	28	4	.17058

One pixel average	-.193
standard deviation	.166
Four and sixteen pixel average	.355
standard deviation	.198

ate the contribution to the horizontal correlations from the scanning process.

The correlations for this data sample were not as widespread as anticipated. This is possibly explained by the fact that the background was clear sky when the digital values were single pixel outputs. During the last segment of the test when four and sixteen pixel sums were used, the background was probably ground clutter as the missile had moved below the horizon. This fact may supply some understanding of why spatial correlations for the last segment seemed equal to or larger than those of the single pixel correlations (i.e. four times the initial correlation distance, or greater). Obviously, more data analysis is needed

TABLE B-IV

Second Vertical Neighbor Correlations
A to A and B to B

Record	Column (A_s or B_s)	No. of Entries per Column	No. of Pixels per Entry	Correlation Coef (r)
114	1(A)	12	1	-.1779
	(B)			.2149
114	2(A)	12	1	.6100
	(B)			.2280
119	1(A)	12	1	.1365
	(B)			.1908
119	2(A)	12	1	.0133
	(B)			-.0987
119	29(A)	12	1	.1030
	(B)			.3381
119	30(A)	12	1	.4717
	(B)			-.3122
181	23(A)	16	4	.664
	(B)	15		.443
181	24(A)	16	4	.2745
	(B)	15		.3287
185	1(A)	15	16	.2961
	(B)			.3598
179	21(A)	14	4	.38596
	(B)			.09963
179	22(A)	14	4	.37803
	(B)			.11464

One pixel average	A .193	B .126	A&B .160
standard deviation	A .294	B .228	A&B .261
Four and sixteen pixel average	A .400	B .269	A&B .335
standard deviation	A .156	B .154	A&B .155

on different kinds of background to gain more insight into the time correlations and their values. However, the spatial correlations of the background noise were substantiated.

Appendix C

Real Data Analysis for Temporal Correlations of Background Noise

Real data was analyzed using REFERENCING, a FORTRAN software package supplied by AFWL, to substantiate the existence of, and to determine numerical values for, the temporal correlations of the background noise of the FLIR pixels. REFERENCING, described in great detail in Reference 12, was written in order to provide a computer simulation capability for the digital correlation trackers using real data values stored on magnetic tape. The data tapes containing the raw data to be read into REFERENCING were supplied by AFWL and were test runs of the FLIR tracking a receding aircraft's jet exhaust. The background was clear sky. Other target scenarios with various backgrounds should also be studied when the data is available. The four data tapes supplied by AFWL were previewed but none were found to have anything other than clear sky backgrounds.

REFERENCING was used as a tool to compute the temporal cross-correlations of a 16-by-16 section of sky between two consecutive data frames (one thirtieth of a second apart). REFERENCING works by starting with a 16-by-16 reference matrix, Q, in the first frame. Assuming that the upper left hand pixel coordinates of Q are (m,n), then the cross-correlations of Q with 256 sixteen-by-sixteen submatrices of a 32-by-32 search matrix in the next frame are computed. Since the upper left-hand coordinates of the search matrix are (m-8, n-8), it uniformly surrounds the reference matrix. REFERENCING was run so that sections of clear sky background FLIR data in one frame (i.e., specifically rejecting target intensity effects) would be searched for the highest cross correlation to the 16-by-16 clear sky background reference data of the preceding frame. The hypothesis is that the same piece of physical background imaged in Q has moved

less than eight pixels (a good assumption based on the target scenario) in one thirtieth of a second, and that the location of this square segment of background can be found in the search matrix based on the maximum cross-correlation. The formula for cross-correlation coefficient computations is derived from the standard statistical formula used to determine correlation coefficients and is (Ref 13:35)

$$r_c = \frac{\mathrm{Cov}[\underline{P},\underline{Q}]}{\sigma_P \sigma_Q} = \frac{E\{\underline{P},\underline{Q}\} - m_P m_Q}{\sigma_P \sigma_Q} \qquad \text{(C-1)}$$

where $\underline{P}$ is a 16-by-16 search submatrix in the frame following that of $\underline{Q}$, and Cov[P,Q] is the covariance of matrices $\underline{P}$ and $\underline{Q}$ which is defined atypically in the following way. The expected value of $\underline{P}$ and $\underline{Q}$, $E\{\underline{P},\underline{Q}\}$, and their separate means, m_P and m_Q, are defined as (Ref 12:36):

$$E\{\underline{P},\underline{Q}\} = \frac{1}{256} \sum_{i=1}^{16} \sum_{j=1}^{16} P_{ij} Q_{ij} \qquad \text{(C-2)}$$

$$m_P = \frac{1}{256} \sum_{i=1}^{16} \sum_{j=1}^{16} P_{ij} \qquad \text{(C-3)}$$

$$m_Q = \frac{1}{256} \sum_{i=1}^{16} \sum_{j=1}^{16} Q_{ij} \qquad \text{(C-4)}$$

The definitions of $E\{\underline{P},\underline{Q}\}$, m_P, m_Q, and, consequently, $\mathrm{Cov}[\underline{P},\underline{Q}]$ are somewhat unusual because they are matrices and not vectors. σ_P and σ_Q, the standard deviations of $\underline{P}$ and $\underline{Q}$, respectively, are computed as follows (Ref 13:36):

$$\sigma_P = \frac{1}{256} \sum_{i=1}^{16} \sum_{j=1}^{16} P_{ij}^2 - m_P^2 \qquad \text{(C-5)}$$

$$\sigma_Q = \frac{1}{256} \sum_{i=1}^{16} \sum_{j=1}^{16} Q_{ij}^2 - m_Q^2 \qquad \text{(C-6)}$$

The results of the cross-correlation runs are shown in Table C-I. The columns of data shown are the mission label of the data tape, the frames compared, the upper left-hand coordinates of the reference and the search matrix, and the maximum cross-correlation coefficient. Table C-I shows three separate runs as separated by the dashed lines. Each run processed five consecutive frames, yielding four maximum cross-correlation coefficients. The starting upper left-hand coordinates of the first reference matrix in the first frame is a program input. Subsequent reference matrices in the same run are centered about the position of the maximum cross-correlation coefficient. Thus, the upper left-hand coordinates of the reference matrices for the second, third, and fourth computations of each run may vary depending on the position of the previous maximum cross-correlation coefficient.

The data shows a consistent positive trend, though it is disappointingly low, averaging .364. The clear sky background has little infrared structure which would remain temporally constant, and so the low cross-correlations are not too surprising. More precise cross-correlation coefficient calculations need to be done on a variety of fixed backgrounds with a motionless sensor. The analysis performed establishes the temporal nature of the specific clear sky background noise and no more.

TABLE C-I

Data Tape	Frames Compared	Start Coordinates (Row, Column)		Maximum Coefficient
MIS 10A	31	25	20	.403
	32	17	12	
MIS 10A	32	30	20	.337
	33	22	12	
MIS 10A	33	34	19	.362
	34	26	11	
MIS 10A	34	39	19	.397
	35	31	11	
MIS 10A	301	25	20	.327
	302	17	12	
MIS 10A	302	30	20	.375
	303	22	12	
MIS 10A	303	25	20	.310
	304	17	12	
MIS 10A	304	30	20	.396
	305	22	12	
MIS 24A	31	25	20	.386
	32	17	12	
MIS 24A	32	20	21	.315
	33	12	13	
MIS 24A	33	19	22	.393
	34	11	14	
MIS 24A	34	23	24	.367
	35	15	16	

Appendix D

Derivation of $\underline{Q}_{Td}$ for the Truth Model, Chapter II

The derivation of $\underline{Q}_{Td}(t_i)$ for the new truth model state equation is an extensive exercise requiring large amounts of integral calculus, matrix multiplications, and algebraic manipulations and requires a detailed description. From Chapter II, $\underline{\Phi}_T(\Delta t)$, $\underline{G}_T$ and $\underline{Q}_T$ are given as

$$\underline{\Phi}_T(\Delta t) = \left[\begin{array}{cc|ccc|ccc} 1 & 0 & & 0 & & & 0 & \\ 0 & 1 & & & & & & \\ \hline & & e^{-a\Delta t} & 0 & 0 & & & \\ & & 0 & e^{-b\Delta t} & \Delta t e^{-b\Delta t} & & 0 & \\ & & 0 & 0 & e^{-b\Delta t} & & & \\ \hline & & & & & e^{-a\Delta t} & 0 & 0 \\ & 0 & & 0 & & 0 & e^{-b\Delta t} & \Delta t e^{-b\Delta t} \\ & & & & & 0 & 0 & e^{-b\Delta t} \end{array}\right] \tag{D-1}$$

$$G_T = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & G_1 & 0 \\ 0 & 0 & G_2 & 0 \\ 0 & 0 & G_3 & 0 \\ 0 & 0 & 0 & G_1 \\ 0 & 0 & 0 & G_2 \\ 0 & 0 & 0 & G_3 \end{bmatrix} \tag{D-2}$$

$$Q_T = \begin{bmatrix} \sigma_D^2 & & & 0 \\ & \sigma_D^2 & & \\ & & 1 & \\ 0 & & & 1 \end{bmatrix} \tag{D-3}$$

where σ_D = the rms value of the dynamics driving noise

$\Delta t = t_{i+1} - t_i$ = sample period

Using these values $\underline{Q}_{Td}$ is evaluated using the following formula (Ref 8:171)

$$\underline{Q}_{Td} = \int_{t_i}^{t_{i+1}} \underline{\Phi}_T(t_{i+1}-\tau)\underline{G}_T(\tau)\underline{Q}_T(\tau)\underline{G}_T^T(\tau)\underline{\Phi}_T^T(t_{i+1}-\tau)d\tau \quad \text{(D-4)}$$

Before actually performing the integral, the matrices must be multiplied out. If the following substitutions are made in (D-1), the multiplications are less confusing:

$$A = e^{-a(t_{i+1}-t_i)} \quad \text{(D-5)}$$

$$B = e^{-b(t_{i+1}-t_i)} \quad \text{(D-6)}$$

$$C = (t_{i+1}-t_i)\, e^{-b(t_{i+1}-t_i)} \quad \text{(D-7)}$$

so that

$$\underline{\Phi}_T = \left[\begin{array}{cc|ccc|ccc} 1 & 0 & & & & & & \\ 0 & 1 & & 0 & & & 0 & \\ \hline & & A & 0 & 0 & & & \\ & & 0 & B & C & & & \\ 0 & & 0 & 0 & B & & 0 & \\ \hline & & & & & A & 0 & 0 \\ 0 & & & 0 & & 0 & B & C \\ & & & & & 0 & 0 & B \end{array}\right] \quad \text{(D-8)}$$

Using $\underline{\Phi}_T(t_{i+1}-t_i)$ from (D-8), $\underline{G}_T$ from (D-2) and $\underline{Q}_T$ as shown in (D-3), then

$$\underline{\Phi}_T\underline{G}_T\underline{Q}_T\underline{G}_T^T\underline{\Phi}_T^T = \left[\begin{array}{cc|ccc|ccc} \sigma_D^2 & 0_2 & & & & & & \\ 0 & \sigma_D^2 & & 0 & & & 0 & \\ \hline & & m_{11} & m_{12} & m_{13} & & & \\ & 0 & m_{12} & m_{22} & m_{23} & & 0 & \\ & & m_{13} & m_{23} & m_{33} & & & \\ \hline & & & & & m_{11} & m_{12} & m_{13} \\ & 0 & & 0 & & m_{12} & m_{22} & m_{23} \\ & & & & & m_{13} & m_{23} & m_{33} \end{array}\right] \quad \text{(D-9)}$$

where

$$m_{11} = A^2G_1^2 \quad \text{(D-10)}$$

$$m_{12} = ABG_1G_2 + ACG_1G_3 \quad \text{(D-11)}$$

$$m_{13} = ABG_1G_3 \quad \text{(D-12)}$$

$$m_{22} = B^2G_2^2 + 2BCG_2G_3 + C^2G_3^2 \quad \text{(D-13)}$$

$$m_{23} = B^2G_2G_3 + BCG_3^2 \quad \text{(D-14)}$$

$$m_{33} = B^2G_3^2 \quad \text{(D-15)}$$

If (D-10) through (D-15) and (D-5) through (D-7) were substituted into (D-9), the resulting matrix would be cumbersome in size and complexity.

Since the integral of the matrix shown in (D-9) equals a matrix of the integrals of each element, the remaining demonstration of $\underline{Q}_{Td}$ computation will follow the integration

of m_{22} of (D-9). Using the expressions for B and C from (D-6) and (D-7), the expression for m_{22} as shown in (D-13) becomes

$$m_{22} = G_2^2 e^{-2b(t_{i+1}-t_i)} + 2G_2G_3(t_{i+1}-t_i)\, e^{-2b(t_{i+1}-t_i)} + G_3^2(t_{i+1}-t_i)^2\, e^{-2b(t_{i+1}-t_i)} \qquad \text{(D-16)}$$

The integration of m_{22} which would supply two terms on the main diagonal of $\underline{Q}_{Td}$ is accomplished as follows:

$$\int_{t_i}^{t_{i+1}} m_{22}(\tau)\,d\tau$$

$$= \int_{t_i}^{t_{i+1}} [G_2^2 e^{-2b(t_{i+1}-\tau)} + 2G_2G_3(t_{i+1}-\tau)\, e^{-2b(t_{i+1}-\tau)} + G_3^2(t_{i+1}-\tau)^2\, e^{-2b(t_{i+1}-\tau)}]\,d\tau \qquad \text{(D-17)}$$

$$= G_2^2 \int_{t_i}^{t_{i+1}} e^{-2b(t_{i+1}-\tau)}\,d\tau + 2G_2G_3 \int_{t_i}^{t_{i+1}} (t_{i+1}-\tau) e^{-2b(t_{i+1}-\tau)}\,d\tau + G_3^2 \int_{t_i}^{t_{i+1}} (t_{i+1}-\tau)^2\, e^{-2b(t_{i+1}-\tau)}\,d\tau$$

After computing the indicated integrals and regrouping the results, the resulting integration of m_{22} is:

$$\int_{t_i}^{t_{i+1}} m_{22}(\tau)\,d\tau$$

$$= (1-e^{-2b(t_{i+1}-t_i)})[G_2^2/2b + 2G_2G_3/(2b)^2 + 2G_3^2/(2b)^3] \qquad \text{(D-18)}$$

$$+ (t_{i+1}-t_i)e^{-2b(t_{i+1}-t_i)}[G_2G_3/b + 2G_3^2/(2b)^2 - (t_{i+1}-t_i)G_3^2/2b]$$

This is the final computed form of m_{22}. An equivalent form of this equation can be found in the computer codes found in Appendix F. The results of the integration of the remaining terms in (D-9) can also be found in Appendix F and will not be shown here.

Appendix E

Computer Simulation Considerations

To understand the detailed test procedure, a discussion of the computer simulation techniques is appropriate. The software was written in FORTRAN IV, and the simulations were accomplished on a CDC 6600. Due to the size of the software and the time required, the scftware was written (as originally organized by Captain Mercier, Ref 5:39) in three large program modules and was run in two steps, to be described herein. The generation of Gaussian noise, the approximations used to produce FLIR values, the Monte Carlo simulation technique, and the tuning of the filter are also clarified in this discussion.

For the first step of the computer simulation, two software modules, called the main program and the library of subroutines, (deck listings can be found in Appendix F) are run together. The library of subroutines (these routines are 90% intact from their original design by Mercier, Ref 5:85-94) is a group of small, flexible programs which usually require infrequent modifications. The main program contains the software for setting up and executing the truth model and filter. Consequently, this module changed considerably and constantly during the truth model evolution and filter design. In its final version the main program required about two dozen parameters, such as truth model and filter target sizes and intensities, and rms noise values, to be read in as data before execution. Besides requiring such inherent math functions as sine, cosine, absolute value, etc., the main program utilizes a matrix inversion routine from the IMSL library which is available at AFIT. The library of subroutines also made use of a pseudo-random number generator.

The third module, the PLOT routines, (the PLOT deck listing is found in Appendix G) is run subsequent to the

execution of the main program and the library of subroutines, which generate and store the data used by the PLOT routines. These programs, which were written by Captain Mercier in 1978, required only small modifications for their usage in this study to generate the plots seen in this Appendix, in Chapters III, V, and VI, and in Appendices I, J, and L.

The Gaussian white noise terms used in the simulation are generated in a subroutine using a pseudo-random number generator with a uniform probability density function between 0 and 1. To produce a discrete realization, v(k) of Gaussian distributed, white noise of a given variance, σ_w, twelve independent values are used from the random number generator in the following equation (Ref 5:41):

$$\underline{v}(k) = \sigma_w \left\{ \sum_{i=1}^{12} u[12(k-1) + i] - 6 \right\} \qquad \text{(E-1)}$$

for $k = 1,2,3,...$

where $u(k)$ = random number from the RAND function.

This equation will produce a white sequence of numbers that is very nearly Gaussian with zero mean and desired variance, as can be argued based on the Central Limit Theorem (Ref 8: 109-110).

Reasonable approximation methods for evaluating the outputs of the FLIR are needed for the computer simulation. The precise mathematical equation would entail evaluating

$$\frac{1}{Ap} \iint_{\substack{\text{i-jth} \\ \text{pixel}}} I_{max} \exp\left\{ -\frac{1}{2} \begin{bmatrix} x-x_{peak} \\ y-y_{peak} \end{bmatrix}^T \underline{P}^{-1} \begin{bmatrix} x-x_{peak} \\ y-y_{peak} \end{bmatrix} \right\} dxdy \qquad \text{(E-2)}$$

where

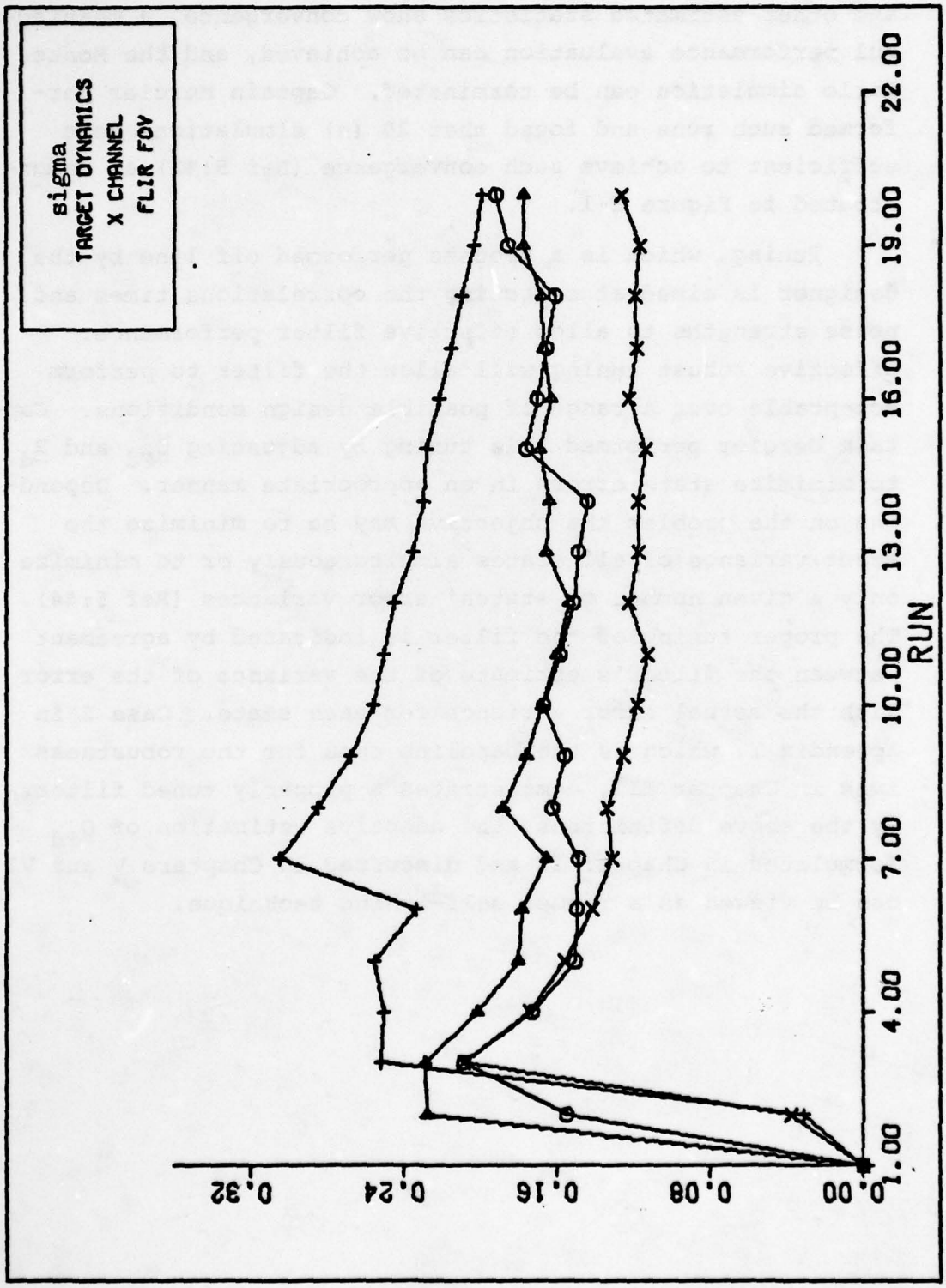
$$\underline{P} = \begin{bmatrix} \sigma_x^2 & r\sigma_x\sigma_y \\ r\sigma_y\sigma_x & \sigma_y^2 \end{bmatrix} \qquad \text{(E-3)}$$

as in equation (7) for the truth model. Similar expressions are evaluated in the filter's estimate of the measurements and in the computation of the partials found in equation (15). Equation (E-2) above represents the average value of the Gaussian intensity function for the pixel. For online usage a reasonable approximation of equation (E-2) is simply to evaluate the intensity function

$$I_{max} \exp\left\{-\frac{1}{2}\begin{bmatrix} x-x_{peak} \\ y-y_{peak} \end{bmatrix}^T \underline{P}^{-1} \begin{bmatrix} x-x_{peak} \\ y-y_{peak} \end{bmatrix}\right\} \qquad \text{(E-4)}$$

at the middle of the pixel, and the filter uses this single point approximation to the integral. This approximation is poor when the eigenvalues of $\underline{P}$ are small compared to the size of a pixel. A more accurate approximation can be computed by dividing the pixel into smaller areas, computing the midpoint value for each of these small areas, and averaging the results. Because the truth model is to depict the real world, an accurate pixel evaluation was performed by using the midpoints of a 4-by-4 array of equal subdivisions of each pixel.

As argued extensively by Captain Mercier on page 41 of his thesis, a Monte Carlo simulation is needed to accomplish a performance evaluation. The inherent goal of Monte Carlo simulation technique is that sufficient numbers of simulations be run so that convergence is reached in the standard deviation (and other estimated statistics) of filter errors, as shown in Figure E-1. A simple covariance analysis of filter performance is not possible for the adaptive extended Kalman filter investigated in this report due to the truth model nonlinearities involved (Ref 8:329). Consequently, a sample-by-sample simulation using white Gaussian noise realizations generated by (E-1) is conducted. The adaptive extended Kalman filter produces state estimates from the available noisy measurements. If, after n number of



SIGMA CONVERGENCE

Figure E-1. Convergence of Standard Deviation

simulations, the standard deviation of the filter errors and other estimated statistics show convergence, a meaningful performance evaluation can be achieved, and the Monte Carlo simulation can be terminated. Captain Mercier performed such runs and found that 20 (n) simulations were sufficient to achieve such convergence (Ref 5:42) as demonstrated in Figure E-1.

Tuning, which is a process performed off line by the designer is aimed at selecting the correlations times and noise strengths to allow effective filter performance. Effective robust tuning will allow the filter to perform acceptable over a range of possible design conditions. Captain Mercier performed this tuning by adjusting $\underline{Q}_{Fd}$ and $\underline{R}_d$ to minimize state errors in an appropriate manner. Depending on the problem the objective may be to minimize the error variance of all states simultaneously or to minimize only a given number of states' error variances (Ref 5:44). The proper tuning of the filter is indicated by agreement between the filter's estimate of the variance of the error with the actual error variance for each state. Case 2 in Appendix I, which is the baseline case for the robustness runs in Chapter III, demonstrates a properly tuned filter. By the above definitions, the adaptive estimation of $\underline{Q}_{Fd}$ formulated in Chapter IV and discussed in Chapters V and VI can be viewed as a robust self-tuning technique.

Appendix F

Main Program and Library Listings

This appendix contains the FORTRAN code for the algorithms presented earlier in the text. Presented first is the eight state filter program in its final form. Next, the six state filter program in its final form is listed. The third listing is of the 72-state temporal background noise, truth model with Mercier's four state filter. Finally, the subroutines used by the eight state program are listed. The eight state filter subroutines cannot be used with the six or four state filter as minor modifications to several subroutines are required for use with the six or four state filter.

8 State Filter

```
      PROGRAM THESIS(INPUT=/80,OUTPUT,TAPE5=INPUT,TAPE6=OUTPUT,TAPE8)
      COMMON/FLIR/ XFOV,YFOV,IMAX,NPIX,SIGMS,SIGMF,SIGMAB,SIGFLR,RF
     *,ASPRO,FIMAX,UT(2,1),IMEAS,AR,XF4(8),SIGVF,PL2P(64,2),VMAX,SIGMFO
     *,RANGE0,RANGE,SIGPVF,CSTH,SNTH
      INTEGER ONE
      REAL IMAX
      DIMENSION Z(64),PHI(8,8),Q(3,3),
     * WORK(3,3),W(8),TEMP(8,8),TEMP1(8,8),SAVE(14),QFD1(8,8),
     *QD(8,8),PHIF(8,8),QFD(8,8),SQQD(8,8),XS(8),H(64,8),PFP(8,8),
     *XFP(8),PFM(8,8),HF(64),EXTRA(8,8),RIH(64,1),DXDXT(8,8),PPFP(8,8),
     *HT(8,64),WKAREA(50,50),C(5),R(64,64),RN(64,64),BD(8,2),PHIFT(8,8),
     *QFDMAX(8),HTZHO(2),DX(8),SIG(2),)(2),PFPOLD(8,8),XFPO(8)
      WRITE(6,1)
    1 FORMAT(1H1)
C
C        READ AND ECHO DATA
C
      READ *, SIGS1
      PRINT *, "RMS DYNAMICS FOR TRUTH MODEL,          SIGS1 = ",SIGS1
      READ *, SIGMAB
      PRINT *, "RMS TRUTH MODEL BACKGROUND NOISE,     SIGMAB = ",SIGMAB
      READ *, SIGFLR
      PRINT *, "RMS TRUTH MODEL FLIR NOISE,           SIGFLR = ",SIGFLR
      READ *, IMAX
      PRINT *, "TRUTH MODEL MAX INTENSITY,              IMAX = ",IMAX
      READ *, SIGAT
      PRINT *, "RMS ATMOSPHERICS FOR TRUTH MODEL,      SIGAT = ",SIGAT
      READ *, NRUN
      PRINT *, "NUMBER OF MONTE CARLO RUNS,             NRUN = ",NRUN
      READ *, TFINAL
      PRINT *, "FINAL TIME,                           TFINAL = ",TFINAL
      READ *,SIGMS
      PRINT *, "INITIAL RMS TRUTH MODEL SIGMA PERVEL, SIGMS = ",SIGMS
      READ *, ASPRO
      PRINT *, "TARGET ASPECT RATIO,                   ASPRO = ",ASPRO
      READ *,X0
      PRINT *, "INITIAL X POSITION,                       X0 = ",X0
      READ *,Y0
      PRINT *, "INITIAL Y POSITION,                       Y0 = ",Y0
      READ *,Z0
      PRINT *, "INITIAL Z POSITION,                       Z0 = ",Z0
      READ *,XDOT0
      PRINT *, "INITIAL VELOCITY, X DIRECTION ,        XDOT0 = ",XDOT0
      READ *,YDOT0
      PRINT *, "INITIAL VELOCITY, Y DIRECTION ,        YDOT0 = ",YDOT0
      READ *,ZDOT0
      PRINT *, "INITIAL VELOCITY, Z DIRECTION ,        ZDOT0 = ",ZDOT0
      READ *,ISPTL
      PRINT *, "SPATIAL NOISE: 1-YES, 0-NO             ISPTL = ",ISPTL
      IF(ISPTL.NE.1) GO TO 2
      READ *,C(1),C(2),C(3),C(4),C(5)
      PRINT *, "SPATIAL NOISE CORRELATION COEFFICIENTS: "
      PRINT *,C(1),C(2),C(3),C(4),C(5)
    2 CONTINUE
      PRINT *," "
      READ *, SIGMF0
      PRINT *, "INITIAL FILTER SIGMA VELOCITY,        SIGMF0 = ",SIGMF0
```

```
      READ *, AR0
      PRINT *, "INITIAL FILTER ASPECT RATIO,          AR0 = ",AR0
      READ *, SIGF2
      PRINT *, "RMS FILTER ATMOSPHERIC NOISE,        SIGF2 = ",SIGF2
      READ *, RF
      PRINT *, "FILTER MEASUREMENT NOISE RMS,          RF = ",RF
      READ *, SIGF10
      PRINT *, "INITIAL RMS DYNAMICS FOR FILTER,    SIGF10 = ",SIGF10
      READ *, FIMAX0
      PRINT *, "FILTER MAX INTENSITY,               FIMAX0 = ",FIMAX0
      READ *,IPRINT
      PRINT *, "PRINT COMMAND,  1 YES, 0 NO         IPRINT = ",IPRINT
C
C        PARAMETER VALUES
C
      YFOV=8.
      NPIX=8
      ATAU1=14.14
      ATAU2=659.5
      FTAU2=1./ATAU1
      DT = 1./30.
      IREF=1
      FIMAX=ABS(FIMAX0)
      SIGVF=ABS(SIGVF0)
      SIGF1=ABS(SIGF10)
      QFDMAX(1)=2.
      QFDMAX(2)=QFDMAX(1)
      QFDMAX(3)=14.
      QFDMAX(4)=QFDMAX(3)
      QFDMAX(5)=20.
      QFDMAX(6)=QFDMAX(5)
      QFDMAX(7)=.5
      QFDMAX(8)=QFDMAX(7)
C
C        INITIALIZE TRUTH MODEL VARIABLES
C
      CALL RANSET(75692)
      ONE = 1
      NPS = 8
      NMS = NPIX**2
      NFS=8
      NFS2=NFS*NFS
      NFSM2=NFS-2
      NIS = 3
      RI=1./RF
      SN=SIGMAB/IMAX
      IF(SN.LE.0.) SN=.001
      SN=1./SN
      RANGE0=SQRT(X0**2+Y0**2+Z0**2)
      AGAIN = .351036534 * SIGAT
      IFILE=6
      DELT = -1.*DT
      DO 5 I=1,8
      BD(I,1)=0.
      BD(I,2)=0.
      DO 5 J=1,8
      QD(I,J) = 0.
```

```
      SQQD(I,J) = 0.
      PHI(I,J) = 0.
    5 CONTINUE
      FACT=(AGAIN**2)*(ATAU1**2)*(ATAU2**4)
      FACT1 = ATAU1-ATAU2
      FACT2 = ATAU1+ATAU2
      FACT3 = 2.*ATAU2
      XFOV=8.
      G1 = FACT/(FACT1**4)
      G2 = FACT/(FACT1**3)
      G3 = FACT/(FACT1**2)
      R1 =  1.- EXP(2.*ATAU1*DELT)
      R2 =  1.- EXP(FACT2*DELT)
      R3 =  1.- EXP(2.*ATAU2*DELT)
      R4 = DT*EXP(DELT*FACT2)
      R5 = DT*EXP(2.*ATAU2*DELT)
C
C        FILL OUT TRUTH MODEL PHI MATRIX.
C        SEE MERCIER'S THESIS FOR DERIVATION
C
      PHI(1,1)= 1.
      PHI(2,2) = PHI(1,1)
      PHI(3,3) = EXP(ATAU1*DELT)
      PHI(4,4) = EXP(ATAU2*DELT)
      PHI(4,5) = DELT*PHI(4,4)
      PHI(5,5) = PHI(4,4)
      PHI(6,6) = PHI(3,3)
      PHI(7,7) = PHI(4,4)
      PHI(7,8) = PHI(4,5)
      PHI(8,8) = PHI(5,5)
      WRITE(6,11)
   11 FORMAT(///2X,"THE TRUTH MODEL STATE TRANSITION MATRIX IS:"/)
      CALL MOUT(PHI,NPS,NPS)
C
C        FILL OUT DISCRETE INPUT MATRIX
C
      BD(1,1)= DT
      BD(2,2)= DT
      WRITE(6,15)
   15 FORMAT(///2X,"THE TRUTH MODEL INPUT MATRIX IS:"/)
      CALL MOUT(BD,NPS,2)
C
C        FILL THE QD MATRIX WITH VALUES USING EXACT INTEGRATION
C        SEE MERCIER'S THESIS FOR DERIVATION
C
      QD(1,1)= SIGS1
      QD(2,2) = QD(1,1)
      QD(3,3) = (G1*R1)/(2.*ATAU1)
      QD(3,4) = R2*(G2/FACT2**2-G1/FACT2)-R4*G2/FACT2
      QD(3,5) = G2*R2/FACT2
      QD(4,3) = QD(3,4)
      QD(4,4) = R3*(G1/FACT3-2.*G2/FACT3**2+2.*G3/FACT3**3)-
     * R5*(G2/ATAU2+G3*DT/FACT3-2.*G3/FACT3**2)
      QD(4,5) = R3*(G3/FACT3**2-G2/FACT3)-R5*G3/FACT3
      QD(5,3) = QD(3,5)
      QD(5,4) = QD(4,5)
      QD(5,5) = R3*G3/FACT3
```

```
      DO 20 I=3,5
      DO 20 J=3,5
      QD(I+3,J+3) = QD(I,J)
      Q(I-2,J-2) = QD(I,J)
   20 CONTINUE
      WRITE(6,30)
   30 FORMAT(///2X,"THE TRUTH MODEL QD MATRIX IS:"/)
      CALL MOUT( QD,NPS,NPS)
C
C        TAKING CHOLESKY SQUARE ROOT OF QD
C
      SQQD(1,1) = SQRT(QD(1,1))
      SQQD(2,2) = SQQD(1,1)
      CALL CHOLESK(Q, WORK, NIS)
      DO 33 I=1,NIS
      DO 33 J=1,NIS
      SQQD(I+2,J+2) = WORK(J,I)
      SQQD(I+5,J+5) = WORK(J,I)
   33 CONTINUE
      WRITE(6,35)
   35 FORMAT(///2X,"THE CHOLESKY SQUARE ROOT OF QD IS:"/)
      CALL MOUT(SQQD,NPS,NPS)
      IF (ISPTL.NE.1) GO TO 41
C
C        SET UP SPATIAL NOISE CORRELATION COEFFICIENT MATRIX
C
      N=64
      M=8
      DO 36 I=1,N
      R(I,I)=1.
      IF (I.GE.64) GO TO 36
      R(I,I+1)=C(1)
      IF (I.GE.63) GO TO 36
      R(I,I+2)=C(3)
      IF (I.GE.57) GO TO 36
      R(I,I+6)=C(4)
      R(I,I+7)=C(2)
      R(I,I+8)=C(1)
      R(I,I+9)=C(2)
      R(I,I+10)=C(4)
      IF (I.GE.49) GO TO 36
      R(I,I+14)=C(5)
      R(I,I+15)=C(4)
      R(I,I+16)=C(3)
      R(I,I+17)=C(4)
      R(I,I+18)=C(5)
   36 CONTINUE
      DO 37 I=1,M
      R(8*I-7,8*I)=0.0
      R(8*I-7,8*I-1)=0.0
      R(8*I-6,8*I)=0.0
      IF (I.GE.8) GO TO 37
      R(8*I,8*I+1)=0.0
      R(8*I,8*I+2)=0.0
      R(8*I-1,8*I+1)=0.0
      R(8*I-7,8*I+7)=0.0
      R(8*I-7,8*I+8)=0.0
```

```
            R(8*I-6,8*I+8)=1.0
            IF (I.GE.7) GO TO 37
            R(8*I,8*I+9)=0.0
            R(8*I,8*I+10)=0.0
            R(8*I-1,8*I+9)=0.0
            IF (I.GE.6) GO TO 37
            R(8*I,8*I+17)=0.0
            R(8*I,8*I+18)=0.0
            R(8*I-1,8*I+17)=0.0
   37       CONTINUE
            DO 38 I=1,N
            L=I+1
            DO 38 J=L,N
            R(J,I)=R(I,J)
   38       CONTINUE
            DO 39 I=1,N
            DO 39 J=1,N
            RN(I,J)=SIGMA3*R(I,J)
   39       CONTINUE
            WRITE(6,44)
    44      FORMAT(///2X,"64 X 64 SPATIAL CORRELATION MATRIX:"/)
            CALL MWRITE (RN,64,64,64)
C
C              COMPUTE THE CHOLESKY SQUARE ROOT OF RN
C
            CALL CHOLESK(RN,R,64)
            WRITE(6,48)
    48      FORMAT(///2X,"THE CHOLESKY SQUARE ROOT OF RN:"/)
            CALL MWRITE (R,64,64,64)
            GO TO 42
   41       DO 43 I=1,64
   43       R(I,I)=1.
   42       CONTINUE
C
C              SET UP FILTER MATRICES.
C
            DO 51 I=1,NFS
            DO 51 J=1,NFS
            PHIF(I,J)=0.
    51      QFD(I,J)=0.
C
C              FILL OUT FILTER PHI MATRIX
C
            PHIF(1,1)=1.
            PHIF(1,3)=DT
            PHIF(1,5)=DT**2/2.
            PHIF(2,2)=PHIF(1,1)
            PHIF(2,4)=DT
            PHIF(2,6)=PHIF(1,5)
            PHIF(3,3)=PHIF(1,1)
            PHIF(3,5)=DT
            PHIF(4,4)=PHIF(1,1)
            PHIF(4,6)=DT
            PHIF(5,5)=1.
            PHIF(6,6)=1.
            PHIF(7,7)=EXP(DELT/FTAU2)
            PHIF(8,8)=PHIF(7,7)
```

```
      DO 56 I=1,NFS
      DO 56 J=1,NFS
 56   PHIFT(I,J)=PHIF(J,I)
C
C        FILL OUT FILTER DISCRETE QFD MATRIX
C        FOR START OF ACQUISITION PHASE
C
      QFD(1,1)=DT**5*SIGF1/20.
      QFD(1,3)=DT**4*SIGF1/8.
      QFD(1,5)=DT**3*SIGF1/6.
      QFD(2,2)=QFD(1,1)
      QFD(2,4)=QFD(1,3)
      QFD(2,6)=QFD(1,5)
      QFD(3,1)=QFD(1,3)
      QFD(3,3)=DT**3*SIGF1/3.
      QFD(3,5)=DT**2*SIGF1/2.
      QFD(4,2)=QFD(3,1)
      QFD(4,4)=QFD(3,3)
      QFD(4,6)=QFD(3,5)
      QFD(5,1)=QFD(1,5)
      QFD(5,3)=QFD(3,5)
      QFD(5,5)=SIGF1*DT
      QFD(6,2)=QFD(1,5)
      QFD(6,4)=QFD(3,5)
      QFD(6,6)=QFD(5,5)
      QFD (7,7) = (SIGF2**2)*(1.-EXP(2.*DELT/FTAU2))
      QFD (8,8)=QFD (7,7)
      WRITE(6,40)
 40   FORMAT(///2X,"THE FILTER STATE TRANSITION MATRIX IS:"/)
      CALL MOUT(PHIF,NFS,NFS)
      WRITE(6,45)
 45   FORMAT(///2X,"THE INITIAL FILTER QD MATRIX IS:"/)
      CALL MOUT( QFD,NFS,NFS)
      PRINT *," "
      PRINT *," "
C
      PRINT *,"********** BEGIN THE MONTE CARLO SIMULATION **********"
C
      DO 99 L=1,NRUN
      TIME = 0.
      XCENTR=0.
      YCENTR=0.
      FIMIN =0.
      IMEAS=0
      FIMAX=ABS(FIMAX0)
      AR=AR0
      SIGVF=ABS(SIGMF0)
      SIGF1=ABS(SIGF10)
      VARY0=ABS(SIGF10)
      TRXXT=200.
      MANIN0=0
C
C        RESET INITIAL COMDITIONS FOR NEW RUN
C
      DO 46 I=1,NPS
      XS(I) = 0.
 46   CONTINUE
```

```
      DO 47 I=1,NFS
      XFP(I) = 0.
      XFM(I)=0.
      DO 47 J=1,NFS
      QFD(I,J)=0.
      PFP(I,J) = 0.
   47 CONTINUE
C
C        PROVIDE FILTER WITH INITIAL VELOCITY AND POSITION
C
      RHOR=(XJ**2+ZJ**2)
      RANGE=(RHOR+YJ**2)
      XFP(3)=(Z0*XDOT0-X0*ZDOT0)/(RHOR*.00002)
      RHOR=SQRT(RHOR)
      XFP(4)=(RHOR*YDOT0-Y0*((X0*XDOT0+Z0*ZDOT0)/RHOR))/(RANGE*.00002)
      IF(IPRINT.NE.1) GO TO 1020
      PRINT *," RHOR=",RHOR," RANGE**2=",RANGE," XFP(3)=",XFP(3),
     *" XFP(4)=",XFP(4)
 1020 CONTINUE
C
C        FILL IN P+ AT TIME 0
C


C
      PFP(1,1)=25.
      PFP(2,2)=PFP(1,1)
      PFP(3,3)=2000.
      PFP(4,4)=PFP(3,3)
      PFP(5,5)=100.
      PFP(6,6)=100.
      PFP(7,7)=.2
      PFP(8,8)=.2
C
C        FILL IN QFD AT TIME 0
C
      QFD(1,1)=DT**5*SIGF1/20.
      QFD(1,3)=DT**4*SIGF1/8.
      QFD(1,5)=DT**3*SIGF1/6.
      QFD(2,2)=QFD(1,1)
      QFD(2,4)=QFD(1,3)
      QFD(2,6)=QFD(1,5)
      QFD(3,1)=QFD(1,3)
      QFD(3,3)=DT**3*SIGF1/3.
      QFD(3,5)=DT**2*SIGF1/2.
      QFD(4,2)=QFD(3,1)
      QFD(4,4)=QFD(3,3)
      QFD(4,6)=QFD(3,5)
      QFD(5,1)=QFD(1,5)
      QFD(5,3)=QFD(3,5)
      QFD(5,5)=SIGF1*DT
      QFD(6,2)=QFD(1,5)
      QFD(6,4)=QFD(3,5)
      QFD(6,6)=QFD(5,5)
      QFD (7,7) = (SIGF2**2)*(1.-EXP(2.*DELT/FTAU2))
      QFD (8,8)=QFD (7,7)
C
```

```
C         TIME LOOP STARTS HERE
C
   50 TIME = TIME + DT
      IF(TIME.GT.TFINAL) GO TO 99
C
C         PERFORM TRUTH MODEL SIMULATION
C
      VTIME=TIME-DT/2.
      XVEH=-500.*TIME+X0
      YVEH=-300.*TIME+Y0
      ZVEH=Z0
      XDOT=-500.
      YDOT=-300.
      ZDOT=0.
      RHOR=(XVEH**2+ZVEH**2)
      RANGE=(RHOR+YVEH**2)
      UT(1,1)=(ZVEH*XDOT-XVEH*ZDOT)/(RHOR*.00002)
      RHOR=SQRT(RHOR)
      UT(2,1)=(RHOR*YDOT-YVEH*((XVEH*XDOT+ZVEH*ZDOT)/RHOR))/(RANGE
     -*.00002)
      RANGE=SQRT(RANGE)
      VMAX=SQRT(XDOT**2+YDOT**2+ZDOT**2)/(RANGE*.00002)
      IF(IPRINT.NE.1) GO TO 1025
      PRINT *," "
      PRINT *,"  TIME = ",TIME
      PRINT *,"XDOT=",XDOT,"  YDOT=",YDOT,"  XVEH=",XVEH,"  YVEH=",YVEH,
     *"  RHOR=",RHOR,"  RANGE=",RANGE
      PRINT *,"UT(1,1)=",UT(1,1),"  UT(2,1)=",UT(2,1),"  VMAX=",VMAX
 1025 CONTINUE
      XCENTR=XCENTR+UT(1,1)*BD(1,1)
      YCENTR=YCENTR+UT(2,1)*BD(2,2)
      CALL NOISE (NPS,W)
      CALL MMPY(TEMP,SQQD,W,NPS,NPS,ONE)
      CALL MMPY(TEMP1,PHI,XS,NPS,NPS,ONE)
      CALL MADD(XS,TEMP,TEMP1,NPS,ONE,ONE)
      CALL MMPY(TEMP1,BD,UT,NPS,2,ONE)
      CALL MADD(XS,XS,TEMP1,NPS,ONE,ONE)
C
      CALL SHIFTA(XFP,XFPO,NFS,NFS)
C         FILTER STATE PROPAGATION.
C
  121 CONTINUE
      CALL MMPY(XFM,PHIF,XFP,NFS,NFS,ONE)
      IF(IPRINT.NE.1) GO TO 1001
      IF(TIME.LT.3.8) GO TO 1001
      PRINT *, " "
      PRINT *,"XS"
      CALL MOUT(XS,NPS,ONE)
      PRINT *, " "
      PRINT *,"XFM"
      CALL MOUT(XFM,NFS,ONE)
      PRINT *," "
      PRINT *," QFD EST"
      CALL MOUT(QFD,NFS,NFS)
 1001 CONTINUE
C
C         FORM CENTROID POSITION AND FILL TRUTH ARRAY
```

```
C
      XPEAK = XS(1) + XS(3) + XS(4)  -(FM(1)
      YPEAK = XS(2) + XS(5) + XS(7)  -(FM(2)
      IF(IPRINT.NE.1) GO TO 1030
      PRINT *,"XPEAK=",XPEAK,"  YPEAK=",YPEAK,"  FIMAX=",FIMAX
 1030 CONTINUE
      IF(ABS(XPEAK).GT.3.*ASPRO*SIGMS) GO TO 101
      IF(ABS(YPEAK).GT.3.*ASPRO*SIGMS) GO TO 102
      CALL MEAS(XPEAK,YPEAK,4,Z,R)
      IF(IPRINT.NE.1) GO TO 1002
      IF(TIME.LT.3.8) GO TO 1002
      PRINT *," "
      PRINT *,"Z"
      CALL MOUT(Z,NPIX,NPIX)
 1002 CONTINUE
C
C        SEARCH FOR FIMAX AND FIMIN
C
      IF(FIMAX0.GT.0.) GO TO 59
      FIMX0=FIMAX
      FIMAX=0.
      FIMIN=0.
      DO 55 I=1,NMS
      IF(Z(I).GT.FIMAX) FIMAX = Z(I)
 55   IF(Z(I).LT.FIMIN) FIMIN = Z(I)
      FIMAX=FIMAX+.14*SIGVF**2-1.52*SIGVF+4.35
      FIMAX= .8*FIMX0+ .2*FIMAX
 59   CONTINUE
C
C        FILTER COVARIANCE PROPAGATION
C
      CALL SHIFTA(PFP,PFPOLD,NFS2,NFS2)
      CALL MMPY(EXTRA,PHIF,PFP,NFS,NFS,NFS)
      CALL MMPY(PPFP,EXTRA,PHIFT,NFS,NFS,NFS)
C        PPFP= PHI * P(TI-1)+ * PHIT
      CALL MADD(PFM,PPFP,QFD,NFS,NFS,ONE)
C        PFM = P(TI-)=PHI * P(TI-1)+ * PHIT + G * Q * GT
      IF(IPRINT.NE.1) GO TO 1008
      IF(TIME.LT.3.8) GO TO 1008
      PRINT *," "
      PRINT *," PFM"
      CALL MOUT(PFM,NFS,NFS)
 1008 CONTINUE
C
C        PERFORM MEASUREMENT UPDATE FOR THE FILTER
C        INVERSE COVARIANCE FORM
C        FORM FILTER CENTROID POSITION AND FILL OUT NON LINEAR
C        SMALL H. CALCULATE PARTIAL SMALL H PARTIAL X
C
      XPEAK = XFM(7)
      YPEAK = XFM(8)
      IF((XFM(3)**2+XFM(4)**2).EQ.0.) XFM(3)=.001
      CALL MEASF(XPEAK,YPEAK,ONE,HF,H)
      IF(IPRINT.NE.1) GO TO 1003
      IF(TIME.LT.3.8) GO TO 1003
      PRINT *," "
      PRINT *,"HF"
```

```
      CALL MOUT(HF,NPIX,NPIX)
 1003 CONTINUE
      DO 60 I=1,NMS
      DO 60 J=1,NFS
      HT(J,I) = H(I,J)
   60 CONTINUE
      CALL MMPY(PFP,HT,H,NFS,NMS,NFS)
C        PFP= HT * H  (R-1 IS SCALAR AND MULTIPLIED LATER)
      IDGT = 0
      CALL LINV2F(PFM,NFS,NFS,EXTRA,IDGT,WKAREA,IER)
C        EXTRA= P(TI-)-1
      DO 65 I=1,NFS
      DO 65 J=1,NFS
      PFP(I,J)=PFP(I,J)*RI + EXTRA(I,J)
   65 CONTINUE
C        PFP= P(TI+)-1= HT * R-1 * H + P(TI-)-1
      IDGT = 0
      CALL LINV2F(PFP,NFS,NFS,EXTRA,IDGT,WKAREA,IER)
      DO 70 I=1,NFS
      DO 70 J=1,NFS
      PFP(I,J) = EXTRA(I,J)
   70 CONTINUE
      DO 130 I=1,NFS
      IF(PFP(I,I).GT.0.) GO TO 130
      PRINT *,"PFP(",I,I,")=",PFP(I,I)," RUN=",L," TIME=",TIME
      PFP(I,I)=PFPOLD(I,I)
  130 CONTINUE
      IF(IPRINT.NE.1) GO TO 1004
      IF(TIME.LT.3.9) GO TO 1004
      PRINT *," "
      PRINT *,"PFP"
      CALL MOUT(PFP,NFS,NFS)
 1004 CONTINUE
C        PFP=P(TI+)
      CALL MADD(RIH,Z,HF,NMS,ONE,NFS)
C        RIH=RESIDUALS= Z-SMALL H
      IF(SIGMF0.GT.0.) GO TO 62
C
C        COMPUTE NEW ESTIMATE OF SIGVF AND AR
C
      DO 58 I=1,NMS
      AR= AR+.001*(RIH(I)*PL2P(I,1))
      SIGVF=SIGVF+.001*(RIH(I)*PL2P(I,2))
   58 CONTINUE
      IF(AR.LE.1.) AR=1.
      IF(SIGVF.LE.0.) SIGVF=SIGMF0
   62 CONTINUE
      CALL MMPY(EXTRA,HT,RIH,NFS,NMS,ONE)
      IF(IPRINT.NE.1) GO TO 1007
      IF(TIME.LT.3.8) GO TO 1007
      PRINT *," "
      PRINT *,"HT X (Z - SMALL H)"
      CALL MOUT(EXTRA,NFS,ONE)
 1007 CONTINUE
C        EXTRA= HT * (Z-SMALL H)
      CALL MMPY(DX,PFP,EXTRA,NFS,NFS,ONE)
C        DX = P(TI+) * HT * (Z - SMALL H)
```

```
      DO 75 I=1,NFS
      DX (I)=DX (I)*RI
   75 CONTINUE
C        DX =DELTA X = P(TI+) * HT * R-1 * (Z - SMALL H)
      DO 78 I=1,NFS
      DO 78 J=1,NFS
      DXDXT(I,J)=DX (I)*DX (J)
   78 CONTINUE
      IF(IPRINT.NE.1) GO TO 1005
      IF(TIME.LT.3.8) GO TO 1005
      PRINT *," "
      PRINT *," DXDXT"
      CALL MOUT(DXDXT,NFS,NFS)
 1005 CONTINUE
C        DXDXT= (X(TI+) - X(TI-)) * (X(TI+) - X(TI-))T
      TRXXTO=TRXXT
      TRXXT=0.
      DO 79 I=1,NFS
   79 TRXXT=TRXXT+DXDXT(I,I)
      IF(TRXXT.LE.10.*TRXXTO) GO TO 112
      IF(MANIND.EQ.1) GO TO 112
      ECC=SQRT(1.-(1./AR**2))
      SIG(1)=SQRT(SIGPVF**2/(1.-(ECC*CSTH)**2))
      SIG(2)=SQRT(SIGPVF**2/(1.-(ECC*SNTH)**2))
      IF(IPRINT.NE.1) GO TO 1040
      PRINT *," SIG(1)=",SIG(1),"  SIG(2)=",SIG(2),"ECC=",ECC
 1040 CONTINUE
      DO 116 I=1,2
      DHTZ=EXTRA(I)
      AL10D=(ALOG10(ABS(DHTZ))+.4651*SIG(I)-3.9)/.82
      D(I)=EXP(2.303*AL10D)
      IF(IPRINT.NE.1) GO TO 1045
      PRINT *,"D(",I,")=",D(I),"  DHTZ= ",DHTZ
 1045 CONTINUE
      XFP(I+4)=(2.*D(I)/DT**2)*DHTZ/ABS(DHTZ)+XFPO(I+4)
  116 CONTINUE
      DO 114 I=1,NFS
      IF(I.EQ.5) GO TO 114
      IF(I.EQ.6) GO TO 114
      XFP(I)=XFPO(I)
  114 CONTINUE
      CALL SHIFTA(PFPOLD,PFP,NFS2,NFS2)
      DO 140 I=3,4
      VFACTOR=10./SQRT(PFP(I,I))
      AFACTOR=300./SQRT(PFP(I+2,I+2))
      DO 139 J=1,NFS
      PFP(I,J)=PFP(I,J)*VFACTOR
      PFP(J,I)=PFP(J,I)*VFACTOR
      PFP(I+2,J)=PFP(I+2,J)*AFACTOR
      PFP(J,I+2)=PFP(J,I+2)*AFACTOR
  139 CONTINUE
  140 CONTINUE
      MANIND=1
      GO TO 121
  112 CONTINUE
      MANIND=0
      TRXXT=.8*TRXXTO+.2*TRXXT
```

```
      CALL MADD(XFP,DX,XFM,NFS,ONE,ONE)
C        XFP=X(TI+)= P(TI+) * R-1 * HT * (Z - SMALL H) + X(TI-
      IF(IFRINT.NE.1) GO TO 1006
      IF(TIME.LT.3.8) GO TO 1006
      PRINT *,"XPEAK=",XPEAK," YPEAK=",YPEAK,"  AR=",AR,
     *"  SIGVF=",SIGVF,"  TRXXT=",TRXXT
      PRINT *, " "
      PRINT *,"XFP"
      CALL MOUT(XFP,NFS,ONE)
 1006 CONTINUE
C
C        ACQUISITION OR ESTIMATION OF QFD
C
      IF(SIGF10.GT.0.) GO TO 74
      CALL SHIFTA(QFD,QFD1,NFS2,NFS2)
      IF(TIME-.19) 72,72,57
   57 IF(TIME-.5) 61,80,80
   61 IF(TRXXT-1000.) 72,77,77
C
C        ACQUISITION SCHEDULE CHANGE OF QFD
C
   72 VARYQ=VARYQ-SIGF1*.064444444
      QFD(1,1)=DT**5*VARYQ/20.
      QFD(1,3)=DT**4*VARYQ/8.
      QFD(1,5)=DT**3*VARYQ/6.
      QFD(2,2)=QFD(1,1)
      QFD(2,4)=QFD(1,3)
      QFD(2,6)=QFD(1,5)
      QFD(3,1)=QFD(1,3)
      QFD(3,3)=DT**3*VARYQ/3.
      QFD(3,5)=DT**2*VARYQ/2.
      QFD(4,2)=QFD(3,1)
      QFD(4,4)=QFD(3,3)
      QFD(4,6)=QFD(3,5)
      QFD(5,1)=QFD(1,5)
      QFD(5,3)=QFD(3,5)
      QFD(5,5)=VARYQ*DT
      QFD(6,2)=QFD(1,5)
      QFD(6,4)=QFD(3,5)
      QFD(6,6)=QFD(5,5)
      QFD (7,7) = (SIGF2**2)*(1.-EXP(2.*DELT/FTAU2))
      QFD (8,8)=QFD (7,7)
      GO TO 74
C
C        ESTIMATION OF QFD
C
   77 PRINT *," "
      PRINT *," ADAPTION STARTED AT ",TIME," SEC."
   80 CALL MADD(QFD,DXDXT,PFP,NFS,NFS,ONE)
      CALL MADD(QFD,QFD,PPFP,NFS,NFS,-1)
C        QFD= DXDXT + P(TI+) - PHI * P(TI-1)+ * PHIT
C
C        BOUNDING QFD
C
      DO 84 I=1,NFS
      QFACTOR=1.
      IF(QFD(I,I).GT.0.) GO TO 86
```

U3

```
      QFACTOR=0.
      QFD(I,I)=.1
      GO TO 85
   86 CONTINUE
      IF(SQRT(QFD(I,I)).GT.QFDMAX(I)) QFACTOR=QFDMAX(I)/SQRT(QFD(I,I))
      IF(QFACTOR.GE.1.) GO TO 84
      QFD(I,I)=QFACTOR**2*QFD(I,I)
   85 CONTINUE
      DO 82 J=1,NFS
      IF(I.EQ.J) GO TO 82
      QFD(I,J)=QFD(I,J)*QFACTOR
   82 CONTINUE
      DO 83 K=1,NFS
      IF(I.EQ.K) GO TO 83
      QFD(K,I)=QFD(K,I)*QFACTOR
   83 CONTINUE
   84 CONTINUE
      DO 73 I=1,NFS
      DO 73 J=1,NFS
   73 QFD(I,J)= .2*QFD(I,J)+.8*QFD1(I,J)
   74 CONTINUE
C
C        WRITE DATA TO FILE TAPE8
C
      SAVE(1) = XS(1)
      SAVE(2) = UT(1,1)
      SAVE(3) = XS(2)
      SAVE(4) = UT(2,1)
      SAVE(5) = XFP(1)
      SAVE(6) = XFP(3)
      SAVE(7) = XFP(2)
      SAVE(8) = XFP(4)
      SAVE(9) = XCENTR
      SAVE(10) = YCENTR
      SAVE(11) = PFP(1,1)
      SAVE(12) = PFP(3,3)
      SAVE(13) = PFP(2,2)
      SAVE(14) = PFP(4,4)
      WRITE(8) SAVE
      GO TO 50
  101 PRINT *, "LOST TRACK, X CHANNEL, MEAS CALLED ",IMEAS," TIMES.
     *RUN ",L
      GO TO 105
  102 PRINT *, "LOST TRACK, Y CHANNEL, MEAS CALLED ",IMEAS," TIMES.
     *RUN ",L
  105 DO 110 J=1,14
  110 SAVE(J)=0.
  106 WRITE(8) SAVE
      TIME=TIME+DT
      IF(TIME.GT.TFINAL) GO TO 99
      GO TO 106
   99 CONTINUE
      STOP "FINISH"
      END
```

```
      PROGRAM THESIS(INPUT=/80,OUTPUT,TAPE5=INPUT,TAPE6=OUTPUT,TAPE8)
      COMMON/FLIR/ XFOV,YFOV,IMAX,NPIX,SIGMS,SIGMF,SIGMAB,SIGFLR,RF
     *,ASPRC,FIMAX,UT(2,1),IMEAS,AR,XFM(6),SIGVF,PL2P(64,2),VMAX,SIGMF0
     *,RANGE0,RANGE
      INTEGER ONE
      REAL IMAX
      DIMENSION Z(64),PHI(8,8),Q(3,3),
     * WORK(3,3),W(3),TEMP(8,8),TEMP1(8,8),SAVE(14),QFD1(6,6),
     *QD(8,8),PHIF(6,6),QFD(6,6),SQQD(8,8),XS(8),H(64,6),PFP(6,6),
     *XFP(6),PFM(6,6),HF(64),EXTRA(6,6),RIH(64,1),CXOXT(6,6),PPFP(6,6),
     *HT(6,64),WKAREA(50,50),C(5),R(64,64),RN(64,64),BD(8,2),PHIFT(6,6),
     *QFDMAX(6)
      WRITE(6,1)
    1 FORMAT(1H1)
C
C        READ AND ECHO DATA
C
      READ *, SIGS1
      PRINT *, "RMS DYNAMICS FOR TRUTH MODEL,          SIGS1 = ",SIGS1
      READ *, SIGMAB
      PRINT *, "RMS TRUTH MODEL BACKGROUND NOISE,     SIGMAB = ",SIGMAB
      READ *, SIGFLR
      PRINT *, "RMS TRUTH MODEL FLIR NOISE,           SIGFLR = ",SIGFLR
      READ *, IMAX
      PRINT *, "TRUTH MODEL MAX INTENSITY,             IMAX = ",IMAX
      READ *, SIGAT
      PRINT *, "RMS ATMOSPHERICS FOR TRUTH MODEL,      SIGAT = ",SIGAT
      READ *, NRUN
      PRINT *, "NUMBER OF MONTE CARLO RUNS,            NRUN = ",NRUN
      READ *, TFINAL
      PRINT *, "FINAL TIME,                           TFINAL = ",TFINAL
      READ *,SIGMS
      PRINT *, "INITIAL RMS TRUTH MODEL SIGMA PERVEL, SIGMS = ",SIGMS
      READ *, ASPRO
      PRINT *, "TARGET ASPECT RATIO,                  ASPRO = ",ASPRO
      READ *,X0
      PRINT *, "INITIAL X POSITION,                      X0 = ",X0
      READ *,Y0
      PRINT *, "INITIAL Y POSITION,                      Y0 = ",Y0
      READ *,Z0
      PRINT *, "INITIAL Z POSITION,                      Z0 = ",Z0
      READ *,XDOT0
      PRINT *, "INITIAL VELOCITY, X DIRECTION ,       XDOT0 = ",XDOT0
      READ *,YDOT0
      PRINT *, "INITIAL VELOCITY, Y DIRECTION ,       YDOT0 = ",YDOT0
      READ *,ZDOT0
      PRINT *, "INITIAL VELOCITY, Z DIRECTION ,       ZDOT0 = ",ZDOT0
      READ *,ISPTL
      PRINT *, "SPATIAL NOISE: 1 YES, 0-NO            ISPTL = ",ISPTL
      IF(ISPTL.NE.1) GO TO 2
      READ *,C(1),C(2),C(3),C(4),C(5)
      PRINT *, "SPATIAL NOISE CORRELATION COEFFICIENTS: "
      PRINT *,C(1),C(2),C(3),C(4),C(5)
    2 CONTINUE
      PRINT *," "
      READ *, SIGMF0
      PRINT *, "INITIAL FILTER SIGMA VELOCITY,       SIGMF0 = ",SIGMF0
```

115

```
      READ *, AR0
      PRINT *, "INITIAL FILTER ASPECT RATIO,          AR0 = ",AR0
      READ *, SIGF2
      PRINT *, "RMS FILTER ATMOSPHERIC NOISE,        SIGF2 = ",SIGF2
      READ *, RF
      PRINT *, "FILTER MEASUREMENT NOISE RMS,          RF = ",RF
      READ *, SIGF10
      PRINT *, "INITIAL RMS DYNAMICS FOR FILTER,     SIGF10 = ",SIGF10
      READ *, FIMAX0
      PRINT *, "FILTER MAX INTENSITY,               FIMAX0 = ",FIMAX0
C
C        PARAMETER VALUES
C
      YFOV=8.
      NPIX=8
      ATAU1=14.14
      ATAU2=659.5
      FTAU2=1./ATAU1
      DT = 1./30.
      IREF=1
      FIMAX=ABS(FIMAX0)
      SIGVF=ABS(SIGMF0)
      SIGF1=ABS(SIGF10)
      QFDMAX(1)=2.
      QFDMAX(2)=QFDMAX(1)
      QFDMAX(3)=400.
      QFDMAX(4)=QFDMAX(3)
      QFDMAX(5)=.5
      QFDMAX(6)=.5
C
C        INITIALIZE TRUTH MODEL VARIABLES
C
      CALL RANSET(75682)
      ONE = 1
      NPS = 8
      NMS = NPIX**2
      NFS=6
      NFS2=NFS*NFS
      NFSM2=NFS-2
      NIS = 3
      RI=1./RF
      SN=SIGMAB/IMAX
      IF(SN.LE.0.) SN=.001
      SN=1./SN
      RANGE0=SQRT(X0**2+Y0**2+Z0**2)
      AGAIN = .351006534 * SIGAT
      IFILE=6
      DELT = -1.*DT
      DO 5 I=1,8
      BO(I,1)=0.
      BO(I,2)=0.
      DO 5 J=1,8
      QQ(I,J) = 0.
      SQQD(I,J) = 0.
      PHI(I,J) = 0.
    5 CONTINUE
      FACT=(AGAIN**2)*(ATAU1**2)*(ATAU2**4)
```

```
      FACT1 = ATAU1-ATAU2
      FACT2 = ATAU1+ATAU2
      FACT3 = 2.*ATAU2
      XFOV=8.
      G1 = FACT/(FACT1**4)
      G2 = FACT/(FACT1**3)
      G3 = FACT/(FACT1**2)
      R1 =  1.- EXP(2.*ATAU1*DELT)
      R2 =  1.- EXP(FACT2*DELT)
      R3 =  1.- EXP(2.*ATAU2*DELT)
      R4 = DT*EXP(DELT*FACT2)
      R5 = DT*EXP(2.*ATAU2*DELT)
C
C        FILL OUT TRUTH MODEL PHI MATRIX.
C        SEE MERCIER'S THESIS FOR DERIVATION
C
      PHI(1,1)= 1.
      PHI(2,2) = PHI(1,1)
      PHI(3,3) = EXP(ATAU1*DELT)
      PHI(4,4) = EXP(ATAU2*DELT)
      PHI(4,5) = DELT*PHI(4,4)
      PHI(5,5) = PHI(4,4)
      PHI(6,6) = PHI(3,3)
      PHI(7,7) = PHI(4,4)
      PHI(7,8) = PHI(4,5)
      PHI(8,8) = PHI(5,5)
      WRITE(6,11)
   11 FORMAT(///2X,"THE TRUTH MODEL STATE TRANSITION MATRIX IS:"/)
      CALL MOUT(PHI,NPS,NPS)
C
C        FILL OUT DISCRETE INPUT MATRIX
C
      BD(1,1)= DT
      BD(2,2)= DT
      WRITE(6,15)
   15 FORMAT(///2X,"THE TRUTH MODEL INPUT MATRIX IS:"/)
      CALL MOUT(BD,NPS,2)
C
C        FILL THE QD MATRIX WITH VALUES USING EXACT INTEGRATION
C        SEE MERCIER'S THESIS FOR DERIVATION
C
      QD(1,1)= SIGS1
      QD(2,2) = QD(1,1)
      QD(3,3) = (G1*R1)/(2.*ATAU1)
      QD(3,4) = R2*(G2/FACT2**2-G1/FACT2)-R4*G2/FACT2
      QD(3,5) = G2*R2/FACT2
      QD(4,3) = QD(3,4)
      QD(4,4) = R3*(G1/FACT3-2.*G2/FACT3**2+2.*G3/FACT3**3)-
     * R5*(G2/ATAU2+G3*DT/FACT3-2.*G3/FACT3**2)
      QD(4,5) = R3*(G3/FACT3**2-G2/FACT3)-R5*G3/FACT3
      QD(5,3) = QD(3,5)
      QD(5,4) = QD(4,5)
      QD(5,5) = R3*G3/FACT3
      DO 20 I=3,5
      DO 20 J=3,5
      QD(I+3,J+3) = QD(I,J)
      Q(I-2,J-2) = QD(I,J)
```

```
   20 CONTINUE
      WRITE(6,30)
   30 FORMAT(///2X,"THE TRUTH MODEL QD MATRIX IS:"/)
      CALL MOUT( QD,NPS,NPS)
C
C        TAKING CHOLESKY SQUARE ROOT OF QD
C
      SQQD(1,1) = SQRT(QD(1,1))
      SQQD(2,2) = SQQD(1,1)
      CALL CHOLESK(2,WORK,NIS)
      DO 33 I=1,NIS
      DO 33 J=1,NIS
      SQQD(I+2,J+2) = WORK(J,I)
      SQQD(I+5,J+5) = WORK(J,I)
   33 CONTINUE
      WRITE(6,35)
   35 FORMAT(///2X,"THE CHOLESKY SQUARE ROOT OF QD IS:"/)
      CALL MOUT(SQQD,NPS,NPS)
      IF (ISPTL.NE.1) GO TO 41
C
C        SET UP SPATIAL NOISE CORRELATION COEFFICIENT MATRIX
C
      N=64
      M=8
      DO 36 I=1,N
      R(I,I)=1.
      IF (I.GE.64) GO TO 36
      R(I,I+1)=C(1)
      IF (I.GE.63) GO TO 36
      R(I,I+2)=C(3)
      IF (I.GE.57) GO TO 36
      R(I,I+6)=C(4)
      R(I,I+7)=C(2)
      R(I,I+8)=C(1)
      R(I,I+9)=C(2)
      R(I,I+10)=C(4)
      IF (I.GE.49) GO TO 36
      R(I,I+14)=C(5)
      R(I,I+15)=C(4)
      R(I,I+16)=C(3)
      R(I,I+17)=C(4)
      R(I,I+18)=C(5)
36    CONTINUE
      DO 37 I=1,M
      R(8*I-7,8*I)=0.0
      R(8*I-7,8*I-1)=0.0
      R(8*I-6,8*I)=0.0
      IF (I.GE.8) GO TO 37
      R(8*I,8*I+1)=0.0
      R(8*I,8*I+2)=0.0
      R(8*I-1,8*I+1)=0.0
      R(8*I-7,8*I+7)=0.0
      R(8*I-7,8*I+8)=0.0
      R(8*I-6,8*I+8)=0.0
      IF (I.GE.7) GO TO 37
      R(8*I,8*I+9)=0.0
      R(8*I,8*I+10)=0.0
```

```
      R(8*I-1,8*I+9)=1.0
      IF (I.GE.6) GO TO 37
      R(8*I,8*I+17)=1.0
      R(8*I,8*I+18)=0.0
      R(8*I-1,8*I+17)=0.0
37    CONTINUE
      DO 38 I=1,N
      L=I+1
      DO 38 J=L,N
      R(J,I)=R(I,J)
38    CONTINUE
      DO 39 I=1,N
      DO 39 J=1,N
      RN(I,J)=SIGMAB*R(I,J)
39    CONTINUE
      WRITE(6,44)
44    FORMAT(///2X,"64 X 64 SPATIAL CORRELATION MATRIX:"/)
      CALL MWRITE (RN,64,64,64)
C
C        COMPUTE THE CHOLESKY SQUARE ROOT OF RN
C
      CALL CHOLESK(RN,R,64)
      WRITE(6,48)
48    FORMAT(///2X,"THE CHOLESKY SQUARE ROOT OF RN:"/)
      CALL MWRITE (R,64,64,64)
      GO TO 42
41    DO 43 I=1,64
43    R(I,I)=1.
42    CONTINUE
C
C        SET UP FILTER MATRICES.
C
      DO 51 I=1,NFS
      DO 51 J=1,NFS
      PHIF(I,J)=0.
51    QFD(I,J)=0.
C
C        FILL OUT FILTER PHI MATRIX
C
      PHIF(1,1)=1.
      PHIF(2,2)=PHIF(1,1)
      PHIF(1,3)=DT
      PHIF(2,4)=DT
      PHIF(3,3)=PHIF(1,1)
      PHIF(4,4)=PHIF(1,1)
      PHIF(5,5)= EXP(DELT/FTAU2)
      PHIF(6,6)=PHIF(5,5)
      DO 56 I=1,NFS
      DO 56 J=1,NFS
56    PHIFT(I,J)=PHIF(J,I)
C
C        FILL OUT FILTER DISCRETE QFD MATRIX
C        FOR START OF ACQUISITION PHASE
C
      QFD(1,1)=DT**3*SIGF1/3.
      QFD (2,2)=QFD (1,1)
      QFD (1,3)=DT**2*SIGF1/2.
```

```
      QFD (3,1)=QFD (1,3)
      QFD (2,4)=QFD (1,3)
      QFD (4,2)=QFD (1,3)
      QFD (3,3)=DT*SIGF1
      QFD (4,4)=QFD (3,3)
      QFD (5,5) = (SIGF2**2)*(1.-EXP(2.*DELT/FTAU2))
      QFD (6,6)=QFD (5,5)
      WRITE(6,40)
   40 FORMAT(///2X,"THE FILTER STATE TRANSITION MATRIX IS:"/)
      CALL MOUT(PHIF,NFS,NFS)
      WRITE(6,45)
   45 FORMAT(///2X,"THE INITIAL FILTER QD MATRIX IS:"/)
      CALL MOUT( QFD,NFS,NFS)
      PRINT *," "
      PRINT *," "
C
      PRINT *,"********** BEGIN THE MONTE CARLO SIMULATION **********"
C
      DO 99 L=1,NRUN
      TIME = 0.
      XCENTR=0.
      YCENTR=0.
      FIMIN =0.
      IMEAS=0
      FIMAX=ABS(FIMAX0)
      AR=ARG
      SIGVF=ABS(SIGVF0)
      SIGF1=ABS(SIGF10)
      VARYQ=ABS(SIGF10)
C
C        RESET INITIAL CONDITIONS FOR NEW RUN
C
      DO 46 I=1,NPS
      XS(I) = 0.
   46 CONTINUE
      DO 47 I=1,NFS
      XFP(I) = 0.
      XFM(I)=0.
      DO 47 J=1,NFS
      QFD(I,J)=0.
      PFP(I,J) = 0.
   47 CONTINUE
C
C        PROVIDE FILTER WITH INITIAL VELOCITY AND POSITION
C
      RHOR=(X0**2+Z0**2)
      RANGE=(RHOR+Y0**2)
      XFP(3)=(Z0*XDOT0-X0*ZDOT0)/(RHOR*.00002)
      RHOR=SQRT(RHOR)
      XFP(4)=(RHOR*YDOT0-Y0*((X0*XDOT0+Z0*ZDOT0)/RHOR))/(RANGE*.00002)
C
C        FILL IN P+ AT TIME 0
C
      PFP(1,1)=25.
      PFP(2,2)=PFP(1,1)
      PFP(3,3)=2000.
      PFP(4,4)=PFP(3,3)
```

```
      PFP(5,5)=.2
      PFP(6,6)=PFP(5,5)
C
C        FILL IN QFD AT TIME 0
C
      QFD(1,1)=DT**3*SIGF1/3.
      QFD (2,2)=QFD (1,1)
      QFD (1,3)=DT**2*SIGF1/2.
      QFD (3,1)=QFD (1,3)
      QFD (2,4)=QFD (1,3)
      QFD (4,2)=QFD (1,3)
      QFD (3,3)=DT*SIGF1
      QFD (4,4)=QFD (3,3)
      QFD (5,5) = (SIGF2**2)*(1.-EXP(2.*DELT/FTAU2))
      QFD (6,6)=QFD (5,5)
C
C        TIME LOOP STARTS HERE
C
   50 TIME = TIME + DT
      IF(TIME.GT.TFINAL) GO TO 99
C
C        PERFORM TRUTH MODEL SIMULATION
C
      VTIME=TIME-DT/2.
      XVEH=-500.*TIME+X0
      YVEH=-300.*TIME+Y0
      ZVEH=Z0
      XDOT=-500.
      YDOT=-300.
      ZDOT=0.
      RHOR=(XVEH**2+ZVEH**2)
      RANGE=(RHOR+YVEH**2)
      UT(1,1)=(ZVEH*XDOT-XVEH*ZDOT)/(RHOR*.00002)
      RHOR=SQRT(RHOR)
      UT(2,1)=(RHOR*YDOT-YVEH*((XVEH*XDOT+ZVEH*ZDOT)/RHOR))/(RANGE
     -*.00002)
      RANGE=SQRT(RANGE)
      VMAX=SQRT(XDOT**2+YDOT**2+ZDOT**2)/(RANGE*.00002)
      XCENTR=XCENTR+UT(1,1)*BD(1,1)
      YCENTR=YCENTR+UT(2,1)*BD(2,2)
      CALL NOISE (NPS,W)
      CALL MMPY(TEMP,SQDD,W,NPS,NPS,ONE)
      CALL MMPY(TEMP1,PHI,XS,NPS,NPS,ONE)
      CALL MADD(XS,TEMP,TEMP1,NPS,ONE,ONE)
      CALL MMPY(TEMP1,BD,UT,NPS,2,ONE)
      CALL MADD(XS,XS,TEMP1,NPS,ONE,ONE)
C
C        FILTER STATE PROPAGATION.
C
      CALL MMPY(XFM,PHIF,XFP,NFS,NFS,ONE)
C
C        FORM CENTROID POSITION AND FILL TRUTH ARRAY
C
      XPEAK = XS(1) + XS(3) + XS(4)  -YFM(1)
      YPEAK = XS(2) + XS(6) + XS(7)  -YFM(2)
      IF(ABS(XPEAK).GT.3.*ASPRO*SIGMS) GO TO 101
      IF(ABS(YPEAK).GT.3.*ASPRO*SIGMS) GO TO 102
```

```
      CALL MEAS(XPEAK,YPEAK,4,Z,R)
C
C        SEARCH FOR FIMAX AND FIMIN
C
      IF(FIMAXO.GT.0.) GO TO 59
      FIMAX=0.
      FIMIN=0.
      DO 55 I=1,NMS
      IF(Z(I).GT.FIMAX) FIMAX = Z(I)
   55 IF(Z(I).LT.FIMIN) FIMIN = Z(I)
   59 CONTINUE
C
C        FILTER COVARIANCE PROPAGATION
C
      CALL MMPY(EXTRA,PHIF,PFP,NFS,NFS,NFS)
      CALL MMPY(PPFP,EXTRA,PHIFT,NFS,NFS,NFS)
C        PPFP= PHI * P(TI-1)+ * PHIT
      CALL MADD(PFM,PPFP,QFD,NFS,NFS,ONE)
C        PFM = P(TI-)=PHI * P(TI-1)+ * PHIT + G * Q * GT
C
C        PERFORM MEASUREMENT UPDATE FOR THE FILTER
C        INVERSE COVARIANCE FORM
C        FORM FILTER CENTROID POSITION AND FILL OUT NON LINEAR
C        SMALL H. CALCULATE PARTIAL SMALL H PARTIAL X
C
      XPEAK = XFM(5)
      YPEAK = XFM(6)
      IF((XFM(3)**2+XFM(4)**2).EQ.0.) XFM(3)=.001
      CALL MEASF(XPEAK,YPEAK,ONE,HF,H)
      DO 60 I=1,NMS
      DO 60 J=1,NFS
      HT(J,I) = H(I,J)
   60 CONTINUE
      CALL MMPY(PFP,HT,H,NFS,NMS,NFS)
C        PFP= HT * H  (R-1 IS SCALAR AND MULTIPLIED LATER)
      IDGT = 0
      CALL LINV2F(PFM,NFS,NFS,EXTRA,IDGT,WKAREA,IER)
C        EXTRA= P(TI-)-1
      DO 65 I=1,NFS
      DO 65 J=1,NFS
      PFP(I,J)=PFP(I,J)*RI + EXTRA(I,J)
   65 CONTINUE
C        PFP= P(TI+)-1= HT * R-1 * H + P(TI-)-1
      IDGT = 0
      CALL LINV2F(PFP,NFS,NFS,EXTRA,IDGT,WKAREA,IER)
      DO 70 I=1,NFS
      DO 70 J=1,NFS
      PFP(I,J) = EXTRA(I,J)
   70 CONTINUE
C        PFP=P(TI+)
      CALL MADD(RIH,Z,HF,NMS,ONE,NFS)
C        RIH=RESIDUALS= Z-SMALL H
      IF(SIGMFO.GT.0.) GO TO 62
C
C        COMPUTE NEW ESTIMATE OF SIGVF AND AR
C
      DO 58 I=1,NMS
```

52

```
      AR= AR+.001*(RIH(I)*PL2P(I,1))
      SIGVF=SIGVF+.001*(RIH(I)*PL2P(I,2))
   58 CONTINUE
      IF(AR.LE.1.) AR=1.
      IF(SIGVF.LE.0.) SIGVF=SIGMF0
   62 CONTINUE
      CALL MMPY(EXTRA,HT,RIH,NFS,NMS,ONE)
C        EXTRA= HT * (Z-SMALL H)
      CALL MMPY(XFP,PFP,EXTRA,NFS,NFS,ONE)
C        XFP= P(TI+) * HT * (Z - SMALL H)
      DO 75 I=1,NFS
      XFP(I)=XFP(I)*RI
   75 CONTINUE
C        XFP=DELTA X = P(TI+) * HT * R-1 * (Z - SMALL H)
      DO 78 I=1,NFS
      DO 78 J=1,NFS
      DXDXT(I,J)=XFP(I)*XFP(J)
   78 CONTINUE
C        DXDXT= (X(TI+) - X(TI-)) * (X(TI+) - X(TI-))T
      TRXXT=0.
      DO 79 I=1,NFS
   79 TRXXT=TRXXT+DXDXT(I,I)
      CALL MADD(XFP,XFP,XFM,NFS,ONE,ONE)
C        XFP=X(TI+)= P(TI+) * R-1 * HT * (Z - SMALL H) + X(TI-)
C
C        ACQUISITION OR ESTIMATION OF QFD
C
      IF(SIGF10.GT.0.) GO TO 74
      CALL SHIFTA(QFD,QFD1,NFS2,NFS2)
      IF(TIME-.19) 72,72,57
   57 IF(TIME-.5) 61,80,80
   61 IF(TRXXT-1000.) 72,77,77
C
C        ACQUISITION SCHEDULE CHANGE OF QFD
C
   72 VARYQ=VARYQ-SIGF1*.0644444444
      QFD(1,1)=DT**3*VARYQ/3.
      QFD(2,2)=QFD(1,1)
      QFD(1,3)=DT**2*VARYQ/2.
      QFD(3,1)=QFD(1,3)
      QFD(2,4)=QFD(1,3)
      QFD(4,2)=QFD(1,3)
      QFD(3,3)=DT*VARYQ
      QFD(4,4)=QFD(3,3)
      QFD(5,5)= (SIGF2**2)*(1.-EXP(2.*DELT/FTAU2))
      QFD(6,6)=QFD(5,5)
      GO TO 74
C
C        ESTIMATION OF QFD
C
   77 PRINT *," "
      PRINT *," ADAPTION STARTED AT ",TIME," SEC."
   80 CALL MADD(QFD,DXDXT,PFP,NFS,NFS,ONE)
      CALL MADD(QFD,QFD,PPFP,NFS,NFS,-1)
C        QFD= DXDXT + P(TI+) - PHI * P(TI-1)+ * PHIT
      DO 73 I=1,NFS
      DO 73 J=1,NFS
```

```
 73   QFD(I,J)= .2*QFD(I,J)+.8*QFD1(I,J)
 74   CONTINUE
C
C        WRITE DATA TO FILE TAPE8
C
      SAVE(1) = XS(1)
      SAVE(2) = UT(1,1)
      SAVE(3) = XS(2)
      SAVE(4) = UT(2,1)
      SAVE(5) = XFP(1)
      SAVE(6) = XFP(3)
      SAVE(7) = XFP(2)
      SAVE(8) = XFP(4)
      SAVE(9) = XCENTR
      SAVE(10) = YCENTR
      SAVE(11) = PFP(1,1)
      SAVE(12) = PFP(3,3)
      SAVE(13) = PFP(2,2)
      SAVE(14) = PFP(4,4)
      WRITE(8) SAVE
      GO TO 50
101   PRINT *, "LOST TRACK, X CHANNEL, MEAS CALLED ",IMEAS," TIMES.
     *RUN ",L
      GO TO 105
102   PRINT *, "LOST TRACK, Y CHANNEL, MEAS CALLED ",IMEAS," TIMES.
     *RUN ",L
105   DO 110 J=1,14
110   SAVE(J)=J.
106   WRITE(8) SAVE
      TIME=TIME+DT
      IF(TIME.GT.TFINAL) GO TO 99
      GO TO 106
 99   CONTINUE
      STOP "FINISH"
      END
```

```
      PROGRAM THESIS(INPUT=/*6,OUTPUT,TAPE5=INPUT,TAPE6=OUTPUT,TAPE8)
      COMMON/FLIR/ XFOV,YFOV,IMAX,NPIX,SIGMS,SIGMF,SIGMAB,SIGFLR,RF
     *,ASPRO,FIMAX,JI(2,1),IMEAS
      INTEGER ONE
      REAL IMAX
      DIMENSION Z(6,6),PHI(72,72),Q(3,3),R4(6,6),TA(2,6),WG4(64,1),
     *WORK(3,3),W(6),TEMP(72,72),TEMP1(72,72),SAVE(1-),W72(72,1),
     *CO(72,72),PHIF(4),QF(4,4),SOOO(72,72),XS(72),H(64,4),FFP(4,4)
     *,XFP(4),PFM(4,4),XFM(4),HF(64),EXTRA(4,4),RIM(64,1),WO(64,1),
     *HT(4,64),WKAREA(36,36),C(5),R(64,64),RN(64,64),BO(72,2)
      READ *, DTAU
      PRINT *, "TRUTH MODEL DYNAMICS CORRELATION TIME, DTAU = ",DTAU
      READ *, SIGS1
      PRINT *, "RMS DYNAMICS FOR TRUTH MODEL,          SIGS1 = ",SIGS1
      READ *, SIGMAB
      PRINT *, "RMS TRUTH MODEL BACKGROUND NOISE,     SIGMAB = ",SIGMAB
      READ *, SIGFLR
      PRINT *, "RMS TRUTH MODEL FLIR NOISE,           SIGFLR = ",SIGFLR
      READ *, IMAX
      PRINT *, "TRUTH MODEL MAX INTENSITY,              IMAX = ",IMAX
      READ *, SN
      PRINT *, "TRUTH MODEL SIGNAL TO NOISE RATIO,        SN = ",SN
      READ *, SDSA
      PRINT *, "RMS DYNAMICS/ATMOSPHERIC NOISE RATIO,   SDSA = ",SDSA
      READ *, NRUN
      PRINT *, "NUMBER OF MONTE CARLO RUNS,             NRUN = ",NRUN
      READ *, TFINAL
      PRINT *, "FINAL TIME,                          TFINAL = ",TFINAL
      READ *, SIGF2
      PRINT *, "RMS FILTER ATMOSPHERIC NOISE,         SIGF2 = ",SIGF2
      READ *,SIGMS
      PRINT *, "INITIAL RMS TRUTH MODEL SIGMA REVEL, SIGMS = ",SIGMS
      READ *, SIGMF
      PRINT *, "RMS FILTER BEAMWIDTH,                 SIGMF = ",SIGMF
      READ *, ICORR
      PRINT *, "CORRELATION. 1 YES, 0 NO,             ICORR = ",ICORR
      READ *, RF
      PRINT *, "FILTER MEASUREMENT NOISE RMS,            RF = ",RF
      READ *, SIGF1
      PRINT *, "RMS DYNAMICS FOR FILTER,              SIGF1 = ",SIGF1
      READ *, FTAU1
      PRINT *, "FILTER DYNAMICS CORRELATION TIME,     FTAU1 = ",FTAU1
      READ *, FIMAX
      PRINT *, "FILTER MAX INTENSITY,                 FIMAX = ",FIMAX
      READ *, ASPRO
      PRINT *, "TARGET ASPECT RATIO,                  ASPRO = ",ASPRO
      READ *, VX0
      PRINT *, "INITIAL VELOCITY IN X DIRECTION,        VX0 = ",VX0
      READ *,VY0
      PRINT *, "INITIAL VELOCITY IN Y DIRECTION,        VY0 = ",VY0
      IF(VX0**2+VY0**2.EQ.0.0) VX0=.001
      READ *, DTIM
      PRINT *, "TRUTH MODEL NOISE CORRELATION TIME,     DTIM = ",DTIM
      READ *,ISPTL
      PRINT *, "SPATIAL NOISE: 1-YES, 0-NO              ISPTL = ",ISPTL
      READ *,C(1),C(2),C(3),C(4),C(5)
      PRINT *, "SPATIAL NOISE CORRELATION COEFFICIENTS: "
```

```
      PRINT *,C(1),C(2),C(3),C(4),C(5)
      XFOV=6.
      YFOV=6.
      NPIX=8
      ATAU1=14.14
      ATAU2=666.6
      FTAU2=1./ATAU1
      DT = 1./30.
      IREF=1
C
C        INITIALIZE TRUTH MODEL VARIABLES
C
      CALL RANSET(12345)
      IX=0
      IY=0
      XM=0.0
      YM=0.0
      ONE = 1
      NPS = 72
      NMS = NPIX**2
      NFS=4
      NIS = 3
      RI=1./RF
      AGAIN = .3515153534 * SIGS1/SDSA
      IFILE=0
      DELT = -1.*DT
      UT(1,1)=0.
      UT(2,1)=0.
      DO 5 I=1,8
      BU(I,1)=0.
      BU(I,2)=0.
      DO 5 J=1,8
      GD(I,J) = 0.
      SQQD(I,J) = 0.
      PHI(I,J) = 0.
    5 CONTINUE
      FACT=(AGAIN**2)*(ATAU1**2)*(ATAU2**4)
      FACT1 = ATAU1-ATAU2
      FACT2 = ATAU1+ATAU2
      FACT3 = 2.*ATAU2
      G1 = FACT/(FACT1**4)
      G2 = FACT/(FACT1**3)
      G3 = FACT/(FACT1**2)
      R1 =  1.- EXP(2.*ATAU1*DELT)
      R2 =  1.- EXP(FACT2*DELT)
      R3 =  1.- EXP(2.*ATAU2*DELT)
      R4 = DT*EXP(DELT*FACT2)
      R5 = DT*EXP(2.*ATAU2*DELT)
      R6=EXP(DELT/UTAU)
      PRINT *, "R6=  ",R6
      PHI(1,1)= 1.
      PHI(2,2) = PHI(1,1)
      PHI(3,3) = EXP(ATAU1*DELT)
      PHI(4,4) = EXP(ATAU2*DELT)
      PHI(5,5) = DELT*PHI(4,4)
      PHI(6,6) = PHI(4,4)
      PHI(7,7) = PHI(3,3)
```

```
      PHI(7,7) = PHI(4,4)
      PHI(7,6) = PHI(4,5)
      PHI(6,6) = PHI(5,5)
      DO 9 I=8,72
    9 PHI(I,I)=R6
      WRITE(6,11)
   11 FORMAT(2X,"THE TRUTH MODEL STATE TRANSITION MATRIX IS:"//)
      PRINT *," "
      BD(1,1)= DT
      BD(2,2)= DT
      WRITE(6,15)
   15 FORMAT(2X,"THE TRUTH MODEL INPUT MATRIX IS:"//)
      CALL MOUT(BD,NP3,2)
C
C        FILL THE QD MATRIX WITH VALUES USING EXACT INTEGRATION
C
      QD(1,1)= SIGS1
      QD(2,2) = QD(1,1)
      QD(3,3) = (G1*R1)/(2.*TAU1)
      QD(3,4) = R2*(G2/FACT2**2+2-G1/FACT2)-R4*G2/FACT2
      QD(3,5) = G2*R2/FACT2
      QD(4,3) = QD(3,4)
      QD(4,4) = R3*(G1/FACT3-2.*G2/FACT3**2+2.*G3/FACT3**3)-
     * R5*(G2/ATAU2+G3*DT/FACT3-2.*G3/FACT3**2)
      QD(4,5) = R3*(G3/FACT3**2-G2/FACT3)-R5*G3/FACT3
      QD(5,3) = QD(3,5)
      QD(5,4) = QD(4,5)
      QD(5,5) = R3*G3/FACT3
      DO 20 I=3,5
      DO 20 J=3,5
      QD(I+3,J+3) = QD(I,J)
      Q(I-2,J-2) = QD(I,J)
   20 CONTINUE
      ZDOUGS=SIGMA8*(1.-R6**2.)
      DO 21 I=6,72
   21 QD(I,I)=ZDOUGS
      WRITE(6,30)
   30 FORMAT(2X,"THE TRUTH MODEL QD MATRIX IS:"//)
      SQQD(1,1) = SQRT(QD(1,1))
      SQQD(2,2) = SQQD(1,1)
      CALL CHOLESK(Q,WORK,NIS)
      DO 33 I=1,NIS
      DO 33 J=1,NIS
      SQQD(I+2,J+2) = WORK(J,I)
      SQQD(I+5,J+5) = WORK(J,I)
   33 CONTINUE
      Y=SQRT(ZDOUGS)
      DO 34 I=6,72
   34 SQQD(I,I)=Y
      WRITE(6,35)
   35 FORMAT(2X,"THE CHOLESKY SQUARE ROOT OF QD IS:"//)
      Y=SQRT(SIGMAR)
      IF (ISPTL.NE.1) GO TO 41
C
C   SET UP SPATIAL NOISE CORRELATION COEFFICIENT MATRIX
C
      N=64
```

```
      M=6
      DO 36 I=1,N
      R(I,I)=1.
      IF (I.GE.6) GO TO 36
      R(I,I+1)=C(1)
      IF (I.GE.5) GO TO 36
      R(I,I+2)=C(3)
      IF (I.GE.4) GO TO 36
      R(I,I+3)=C(4)
      R(I,I+7)=C(2)
      R(I,I+6)=C(1)
      R(I,I+5)=C(2)
      R(I,I+10)=C(4)
      IF (I.GE.3) GO TO 36
      R(I,I+14)=C(3)
      R(I,I+15)=C(4)
      R(I,I+16)=C(3)
      R(I,I+17)=C(4)
      R(I,I+18)=C(5)
35    CONTINUE
      DO 37 I=1,M
      R(6*I-7,6*I)=0.0
      R(6*I-6,6*I-1)=0.0
      R(6*I-5,6*I)=0.0
      IF (I.GE.6) GO TO 3
      R(6*I,6*I+1)=0.0
      R(6*I,6*I+2)=0.0
      R(6*I-1,6*I+1)=0.0
      R(6*I-7,6*I+7)=0.0
      R(6*I-6,6*I+6)=0.0
      R(6*I-5,6*I+5)=0.0
      IF (I.GE.7) GO TO 3
      R(6*I,6*I+3)=0.0
      R(6*I,6*I+10)=0.0
      R(6*I-1,6*I+9)=0.0
      IF (I.GE.8) GO TO 37
      R(6*I,6*I+17)=0.0
      R(6*I,6*I+16)=0.0
      R(6*I-1,6*I+17)=0.0
37    CONTINUE
      DO 38 I=1,N
      L=I+1
      DO 38 J=L,N
      R(J,I)=R(I,J)
38    CONTINUE
      DO 39 I=1,N
      DO 39 J=1,N
      RN(I,J)=Y*R(I,J)
39    CONTINUE
      PRINT *," "
      PRINT *,"64 BY 64 COVARIANCE MATRIX:"
      PRINT *," "
      PRINT *," "
C
C     COMPUTE THE CHOLESKI SUCARE ROOOT OF RN
C
      CALL CHOLESK(RN,R,64)
```

```
      GO TO 42
   41 DO 43 I=1,64
   43 R(I,I)=Y
   42 CONTINUE
      PRINT *," "
      PRINT *,"THE CHOLESKI SQUARE ROOT OF RN:"
      PRINT *," "
      PRINT *," "
C
C        SET UP FILTER TIME INVARIANT MATRICES.
C
      PHIF(1) = EXP(DELT/FTAU1)
      PHIF(2) = PHIF(1)
      PHIF(3) = EXP(DELT/FTAU2)
      PHIF(4) = PHIF(3)
      QFD(1) = (SIGF1**2)*(1.-EXP(2.*DELT/FTAU1))
      QFD(2) = QFD(1)
      QFD(3) = (SIGF2**2)*(1.-EXP(2.*DELT/FTAU2))
      QFD(4) = QFD(3)
      WRITE(6,43)
   43 FORMAT(2X,"THE FILTER STATE TRANSITION MATRIX DIAGONAL TERMS ARE")
      CALL MOUT(PHIF,NFS,ONE)
      WRITE(6,45)
   45 FORMAT(2X,"THE FILTER QD MATRIX DIAGONAL TERMS ARE:"//)
      CALL MOUT( QFD,NFS,ONE)
C
      PRINT *,"********** BEGIN THE MONTE CARLO SIMULATION *****"
C
      DO 99 L=1,NRUN
      TIME = 0.
      IFLAG = 0
      YNEW=0.
      XNEW=0.
      IUPDAT=0
      XCENTR=0.
      YCENTR=0.
      XDSUM=0.
      YDSUM=0.
C
C        RESET INITIAL CONDITIONS FOR NEW RUN
C
      DO 46 I=1,NFS
      XS(I) = 0.
   46 CONTINUE
      DO 47 I=1,NFS
      XFF(I) = 0.
      DO 47 J=1,NFS
      PFF(I,J) = 0.
      IF(I.EQ.J) PFF(I,J) = .5
   47 CONTINUE
      CALL NOISE(0.,WD)
      CALL MMPY(WD,R,WE,64,64,1)
      DO 48 I=1,64
   48 XS(64+1)=SIGMAB*WD(I)
      IF(ICORR.EQ.1)CALL MEAS (XNEW,YNEW,NFS,RA,R)
   51 TIME = TIME + DT
      IF(TIME.GT.TFINAL) GO TO 99
```

```
      IFLAG = IFLAG+1
C
C        PERFORM TRUTH MODEL SIMULATION
C
      AR=(DT/2.)*TIME
      UT(1,1)= VX0*EXP(AR)
      UT(2,1)=VY0*EXP(AR/2.)
      XCENTR=XCENTR+UT(1,1)*PD(1,1)
      YCENTR=YCENTR+UT(2,1)*PD(2,2)
      CALL NOISE (Q,W)
      CALL MCASE (Q+,WSQ)
      CALL MMPY (WD,R,WSQ,64,64,1)
      DO 52 I=1,8
52    W72(I,1)=W(I)
      DO 53 I=1,64
53    W72(I+8,1)=WSQ(I,1)
      CALL MMPY(TEMP,3000,W72,NPS,NPS,ONE)
      CALL MMPY(TEMP1,PHI,XS,NPS,NPS,ONE)
      XU=UT*UT(1,1)
      YU=UT*UT(2,1)
      CALL MADD(XS,TEMP,TEMP1,NPS,ONE,ONE)
      CALL MMPY(TEMP1,GD,UT,NPS,2,ONE)
      CALL MADD(XS,XS,TEMP1,NPS,ONE,ONE)
      CALL PIXRO(XU,YU,XM,YM,IX,IY)
      CALL STATMO(XS,IX,IY)
      CALL FILL(XS,IX,IY,SIGMAB)
      XPEAK = XS(1) + XS(3) + XS(4) -XJSUM
      YPEAK = XS(2) + XS(5) + XS(7) -YJSUM
      IF (ABS(XPEAK).GT.3.*ASPRO*SIGMS) GO TO 101
      IF (ABS(YPEAK).GT.3.*ASPRO*SIGMS) GO TO 102
      DO 51 I=1,64
51    Z(I)=XS(I+8)
      CALL MEAS(XPEAK,YPEAK,NFS,Z,K)
C
C        PERFORM CORRELATION TRACKING
C
      IF(ICORR.NE.1)GO TO 5000
      IUPDAT=IUPDAT+1
C
C         FILL THE TARGET ARRAY
C
      CALL SHIFTA(Z,TA,NMS,NPS)
      CALL CORR(RA,TA,NPS,NPS,XCORR,YCORR,DX,DY)
      XCENTR=XNEW+XCORR
      YCENTR=YNEW+YCORR
      IF(IUPDAT.NE.IREF)GO TO 5000
      CALL SHIFTA(Z,RA,NMS,NMS)
      XNEW=XCENTR
      YNEW=YCENTR
      IUPDAT=0
5000  CONTINUE
C
C        PERFORM FILTER STATE AND COVARIANCE PROPOGATION
C
      XFP(1)=0.
      XFP(2)=0.
      DO 61 I=1,NFS
```

```
      DO 55 J=1,NFS
      XFM(I) = PHIF(I)*XFP(I)
      PFM(I,J) = PFP(I,J)*PHIF(I)*PHIF(J)
      IF(I.EQ.J) PFM(I,J) = PFM(I,J) + SPJ(I)
   55 CONTINUE
C
C        PERFORM MEASUREMENT UPDATE FOR THE FILTER
C
      XPEAK = XFM(1) + XFM(3)
      YPEAK = XFM(2) + XFM(4)
      CALL MEASF(XPEAK,YPEAK,ONE,HF,H)
      DO 65 I=1,NMS
      DO 65 J=1,NFS
      HT(J,I) = H(I,J)
   65 CONTINUE
      CALL MMPY(PFP,HT,H,NFS,NMS,NFS)
      IDGT = 0
      CALL LINV2F(PFM,NFS,NFS,EXTRA,IDGT,WKAREA,IER)
      DO 66 I=1,NFS
      DO 66 J=1,NFS
      PFM(I,J)=PFP(I,J)*RI + EXTRA(I,J)
   66 CONTINUE
      IDGT = 0
      CALL LINV2F(PFP,NFS,NFS,EXTRA,IDGT,WKAREA,IER)
      DO 70 I=1,NFS
      DO 70 J=1,NFS
      PFP(I,J) = EXTRA(I,J)
   70 CONTINUE
      CALL MADD(RIH,Z,HF,NMS,ONE,NFS)
      CALL MMPY(EXTRA,HT,RIH,NFS,NMS,ONE)
      CALL MMPY(XFP,PFP,EXTRA,NFS,NFS,ONE)
      DO 75 I=1,NFS
      XFP(I) = XFP(I)*RI + XFM(I)
   75 CONTINUE
      XDSUM=XDSUM+XFP(1)
      YDSUM=YDSUM+XFP(2)
      SAVE(1) = XS(1)
      SAVE(2) = XS(3) + XS(4)
      SAVE(3) = XS(2)
      SAVE(4) = XS(5) + XS(7)
      SAVE(5) = XDSUM
      SAVE(6) = XFP(3)
      SAVE(7) = YDSUM
      SAVE(8) = XFP(4)
      SAVE(9) = XCENTR
      SAVE(10) = YCENTR
      SAVE(11) = PFP(1,1)
      SAVE(12) = PFP(3,3)
      SAVE(13) = PFP(2,2)
      SAVE(14) = PFP(4,4)
      WRITE(8) SAVE
      GO TO 51
  101 PRINT *,"LOST TRACK, X CHANNEL, RUN ",L
      GO TO 103
  102 PRINT *,"LOST TRACK, Y CHANNEL, RUN ",L
  103 DO 110 J=1,14
  110 SAVE(J)=0.
```

```
100   WRITE(6) SAVE
      TIME=TIME+DT
      IF (TIME.GT.TFINAL) GO TO 99
      GO TO 100
   99 CONTINUE
      STOP" FINISH"
      END
```

```
      SUBROUTINE MEAS (XPEAK,YPEAK,ISUB,Z,R)
      COMMON/FLIR/ XFOV,YFOV,IMAX,NPIX,SIGMS,SIGMF,SIGMAB,SIGFLR,RF
     *,ASPRO,FIMAX,UT(2,1),IMEAS,AR,XF4(8),SIGVF,PL2P(64,2),VMAX,SIGMF
     *,RANGE0,RANGE,SIGPVF,CSTH,SNTH
      REAL IMAX
      DIMENSION Z(8,8), R(64,64) ,W1(64,1),WD(64,1)
      ZMIN = 0.
      SIGPV=SIGMS*RANGE0/RANGE
      PLVEL=SQRT(UT(1,1)**2+UT(2,1)**2)
      SNTH=UT(2,1)/PLVEL
      CSTH=UT(1,1)/PLVEL
      SIGV=(1.+(ASPRO-1.)*PLVEL/VMAX)*SIGPV
      I = (NPIX*ISUB)
      IDIV = ISUB**2
      XINCR = XFOV/FLOAT(I)
      YINCR = YFOV/FLOAT(I)
      X = -1.*XFOV/2. + XINCR/2.
      Y = YFOV/2. - YINCR/2.
      X0 = X
      CALL NOISE(64,W1)
      CALL MMPY(WD,R,W1,64,64,1)
      MN=0
      DO 20 K=1,NPIX
      DO 15 J=1,NPIX
      MN=MN+1
      TOTAL = 0.
      XN = X
      YN = Y
      DO 10 N=1,ISUB
      YCSTH=(YN-YPEAK)*CSTH
      YSNTH=(YN-YPEAK)*SNTH
      DO 5 M =1,ISUB
      ARGSPV=YCSTH-(XN-XPEAK)*SNTH
      ARGSV=(XN-XPEAK)*CSTH+YSNTH
      ARG=-((ARGSV/SIGV)**2+(ARGSPV/SIGPV)**2)*.5
      TOTAL =TOTAL+EXP(ARG)*IMAX
      XN = X +FLOAT(M)*XINCR
    5 CONTINUE
      YN = Y- FLOAT(N)*YINCR
      XN = X
   10 CONTINUE
      Z(K,J)=TOTAL/FLOAT(IDIV)
C
C        ADD BACKGROUND AND FLIR NOISE BOTH ZERO MEAN
C
      IF(SIGFLR.EQ.0.) GO TO 30
      GAUSS=0.
      DO 110 LL=1,12
      GAUSS=GAUSS+RANF(DUM1)
  110 CONTINUE
      F=(GAUSS-6.)*SIGFLR
      Z(K,J) = Z(K,J) + F
   30 Z(K,J)=Z(K,J)+WD(MN,1)
      IF(Z(K,J).LT.ZMIN)ZMIN=Z(K,J)
      X = XN +FLOAT(ISUB)*XINCR
   15 CONTINUE
      Y = Y - FLOAT(ISUB)*YINCR
```

```
                X=X0
             20 CONTINUE
                IMEAS=IMEAS+1
                IF(ZMIN.EQ.0.)RETURN
                DO 25 I=1,NPIX
                DO 25 J=1,NPIX
                Z(I,J) = Z(I,J)-ZMIN+.1
             25 CONTINUE
                RETURN
                END
```

SUBROUTINE MMPY 74/74 OPT=1 FTN 4.7+476

```
      1         SUBROUTINE MMPY(C,A,B,K,M,N)
                DIMENSION C(K,N),A(K,M),B(M,N)
                DO 1 I=1,K
                DO 1 J=1,N
                C(I,J)=0.
              1 CONTINUE
                DO 5 L=1,K
                DO 5 J=1,N
                DO 5 I=1,M
                C(L,J) = C(L,J) + (A(L,I)*B(I,J))
              5 CONTINUE
                RETURN
                END
```

SUBROUTINE NOISE 74/74 OPT=1 FTN 4.7+476

```
      1         SUBROUTINE NOISE(N,W)
                DIMENSION W(N)
                DO 15 J=1,N
                TOTAL=0.
      5         DO 5 I=1,12
                TOTAL =TOTAL + RANF(DUM)
              5 CONTINUE
                W(J) = TOTAL - 6.
             15 CONTINUE
     10         RETURN
                END
```

```
      SUBROUTINE MEASF(XPEAK,YPEAK,ISUB,Z,H)
      COMMON/FLIR/ XFOV,YFOV,IMAX,NPIX,SIGMS,SIGMF,SIGMAB,SIGFLR,RF
     *,ASPRO,FIMAX,UT(2,1),IMEAS,AR,XFM(8),SIGVF,PL2P(64,2),VMAX,SIGMF
     *,RANGEC,RANGE,SIGPVF,CSTH,SNTH
      DIMENSION Z(8,8),H(64,8)
      REAL IMAX
      ZMIN = 0.
      PLVEL= SQRT(XFM(3)**2+XFM(4)**2)
      SNTH= XFM(4)/PLVEL
      CSTH= XFM(3)/PLVEL
      SIGPVF=SIGVF/AR
      I = (NPIX*ISUB)
      IDIV = ISUB**2
      XINCR = XFOV/FLOAT(I)
      YINCR = YFOV/FLOAT(I)
      X = -1.*XFOV/2. + XINCR/2.
      Y = YFOV/2. - YINCR/2.
      XB = X
      DO 20 K=1,NPIX
      NUM = K
      DO 15 J=1,NPIX
      TOTAL = 0.
      SUM1=0.
      SUM2=0.
      XN = X
      YN = Y
      DO 10 N=1,ISUB
      YCSTH=(YN-YPEAK)*CSTH
      YSNTH=(YN-YPEAK)*SNTH
      DO 5 M =1,ISUB
      ARGSPV=YCSTH-(XN-XPEAK)*SNTH
      ARGSV=(XN-XPEAK)*CSTH+YSNTH
      ARG=-((ARGSV/SIGVF)**2+(ARGSPV/SIGPVF)**2)*.5
      PART= EXP(ARG)*FIMAX
      TOTAL = TOTAL+PART
      SUM1 = SUM1 + PART*(ARGSV*CSTH/SIGVF**2-ARGSPV*SNTH/SIGPVF**2)
      SUM2 = SUM2 + PART*(ARGSPV*CSTH/SIGPVF**2+ARGSV*SNTH/SIGVF**2)
      XN = X +FLOAT(M)*XINCR
    5 CONTINUE
      YN = Y-FLOAT(N) * YINCR
      XN = X
   10 CONTINUE
      Z(K,J) = TOTAL/FLOAT(IDIV)
      IF(SIGMF.GT.J.) GO TO 16
      PL2P(K+(J-1)*8,1)= PART*(-(ARGSPV/SIGVF)**2*AR)
      PL2P(K+(J-1)*8,2)= PART*((ARGSV**2+ARGSPV**2*AR**2)/SIGVF**3)
   16 CONTINUE
      IF(Z(K,J).LT.ZMIN)ZMIN=Z(K,J)
      H(NUM,1) = SUM1/FLOAT(IDIV)
      H(NUM,2) = SUM2/FLOAT(IDIV)
      H(NUM,3)=0.
      H(NUM,4)=0.
      H(NUM,5)=0.
      H(NUM,6)=0.
      H(NUM,7)=H(NUM,1)
      H(NUM,8)=H(NUM,2)
      X = XN +FLOAT(ISUB)*XINCR
```

```
                NUM = NUM + NPIX
             15 CONTINUE
50              Y = Y - FLOAT(ISUB)*YINCR
                X=X0
             20 CONTINUE
                IF(ZMIN.EQ.0.)RETURN
                DO 25 I=1,NPIX
55              DO 25 J=1,NPIX
                Z(I,J) = Z(I,J)-ZMIN+.1
             25 CONTINUE
                RETURN
                END
```

```
 SUBROUTINE MOUT        74/74   OPT=1                          FTN 4.7+476

 1                  SUBROUTINE MOUT(A,NR,NC)
                    DIMENSION A(NR,NC)
                    DO 10 I=1,NR
                    WRITE(6,5) (A(I,J),J=1,NC)
 5                5 FORMAT(2X,8(G13.7,3X))
                  10 CONTINUE
                    RETURN
                    END
```

```
 SUBROUTINE MADD        74/74   OPT=1                          FTN 4.7+476

 1                  SUBROUTINE MADD(C,A,B,J,K,IFLAG)
                    DIMENSION A(J,K),B(J,K),C(J,K)
                    IF(IFLAG.EQ.1) GO TO 6
                    DO 5 N=1,J
 5                  DO 5 M=1,K
                    C(N,M) = A(N,M) - B(N,M)
                  5 CONTINUE
                    RETURN
                  6 DO 10 N=1,J
10                  DO 10 M=1,K
                    C(N,M) = A(N,M) + B(N,M)
                  10 CONTINUE
                    RETURN
                    END
```

```
      SUBROUTINE CHOLESK(A,S,N)
      DIMENSION A(1),S(1)
      NDIM=N
      NDIM1=N+1
      TOL=1.E-6
      MR=0
      NN=N*NDIM
      TOL1=0.
      DO 1 I=1,NN,NDIM1
      R=ABS(A(I))
    1 IF(R.GT.TOL1)TOL1=R
      TOL1 = TOL1*1.E-12
      II=1
      DO 50 I=1,N
      IM1 = I-1
      DO 5 JJ=I,NN,NDIM
    5 S(JJ) = 0.
      ID = II+IM1
      R=A(ID)-DOT(IM1,S(II),S(II))
      IF(ABS(R).LT.(TOL*A(ID)+TOL1)) GO TO 50
      IF(R) 15,50,20
   15 MR=-1
      WRITE(6,1000)
 1000 FORMAT(37H TRIED TO FACTOR AN INDEFINATE MATRIX    )
      RETURN
   20 S(ID) = SQRT(R)
      MR=MR+1
      IF(I.EQ.N) RETURN
      L=II+NDIM
      DO 25 JJ=L,NN,NDIM
      IJ=JJ+IM1
   25 S(IJ) = (A(IJ)-DOT(IM1,S(II),S(JJ)))/S(ID)
   50 II=II+NDIM
      RETURN
      END
```

```
   FJNCTION DOT         74/74   OPT=1                          FTN 4.7+476
(
1                FUNCTION DOT(NR,A,B)
                 DIMENSION A(1),B(1)
                 DOT=0.
                 DO 1 I=1,NR
5              1 DOT = DOT+A(I)*B(I)
                 RETURN
                 END
```

```
 SUBROUTINE SHIFTA     74/74   OPT=1                          FTN 4.7+476

1                SUBROUTINE SHIFTA(A,B,M,N)
                 DIMENSION A(N),B(N)
                 DO 100 K=1,M
             100 B(K) = A(K)
5                RETURN
                 END
(
```

```
SUBROUTINE RNDF       74/74   OPT=1                          FTN 4.7+476

                 SUBROUTINE RNDF(A,K)
           C        SUBROUTINE TO ROUNDOFF A,TO NEAREST INTEGER,K
           C
                 K=IFIX(A)
                 B=A-K
                 IF(B.LT.0.5)GO TO 100
                 K=K+1
             100 RETURN
                 END
```

```
SUBROUTINE MWRITE     74/74   OPT=1                                  FTN 4.7+478

1                     SUBROUTINE MWRITE (I,N,M,IDIM)
                      DIMENSION A(IDIM,1)
                      PRINT *," "
                      DO 350 I=1,N
5                     WRITE (6,340) (A(I,J),J=1,M)
              340     FORMAT (1X,15(F7.4,1X))
              350     CONTINUE
                      PRINT *," "
                      END


SUBROUTINE FILL       74/74   OPT=1                                  FTN 4.7+478

1                     SUBROUTINE FILL (XS,IX,IY,SIGMA2)
                      DIMENSION XS(1),W(8)
                      R=FLOAT(IX)
                      S=FLOAT(IY)
5                     N=ABS(R)
                      M=ABS(S)
                      IF (IX) 5,10,15
              5       DO 6 J=1,N
                       CALL NOISE(8,W)
0                     DO 6 I=1,8
              6       XS(8*I+J)=SIGMA2*W(I)
                      GO TO 10
              15      DO 7 I=1,N
                       CALL NOISE(8,W)
5                     L=9-I
                      DO 7 J=1,8
              7       XS(8*J+L)=SIG*A2*W(J)
              10      CONTINUE
                      L=0
1                     IF (IY) 20,25,30
              20      DO 21 I=1,M
                      L=9-I
                      CALL NOISE(8,W)
                      DO 21 J=1,8
5             21      XS(8*L+J)=SIG*A2*W(J)
                      GO TO 25
              30      DO 26 I=1,M
                      CALL NOISE(8,W)
                      DO 26 J=1,8
0             26      XS(8*I+J)=SIG*A2*W(J)
              25      CONTINUE
                      RETURN
                      END
```

```
 SUBROUTINE STATMO     74/74   OPT=1                                FTN 4.7+.7

                      SUBROUTINE STATMO (XS,IX,IY)
                      DIMENSION XS(1)
                      M=1
                      R=FLOAT(IX)
                      K=8-ABS(R)
                      DO 10 I=1,8
                      N=8*I
                      DO 10 J=1,K
                      IF (IX) 5,6,6
                5     M=9-J
                      XS(N+M)=XS(N+1-IY)
                      GO TO 10
                6     M=J
                      XS(N+M)=XS(N+1+IX)
               10     CONTINUE
                      S=FLOAT(IY)
                      N=8-ABS(S)
                      M=1
                      DO 21 I=1,N
                      DO 21 J=1,8
                      IF (IY) 16,16,17
               15     M=9-I
                      K=8*M
                      XS(K+J)=XS(K+J-8*IY)
                      GO TO 21
               16     M=I
                      K=8*M
                      XS(K+J)=XS(K+J+8*IY)
               21      CONTINUE
                      RETURN
                      END


 SUBROUTINE PIXMO      74/74   OPT=1                                FTN 4.7+.7

                      SUBROUTINE PIXMO (DX,DY,XM,YM,IX,IY)
                C  GIVEN DX AND DY IN PIXELS/TIME UNIT
                C   ESTIMATE X AND Y VELOCITY IN INTEGER NO. OF PIXELS
                      IX=0
                      IY=0
                      XM=XM+DX
                      YM=YM+DY
                      T1=ABS(XM)
                      T2=ABS(YM)
                      IF (T1 .GE. .3) GO TO 1
                      GO TO 2
                1     CALL RNDF (XM,IX)
                2     XM = XM-FLOAT(IX)
                      IF (T2 .GE. .5) GO TO 3
                      GO TO 4
                3     CALL RNDF(YM,IY)
                4     YM=YM-FLOAT(IY)
                      RETURN
                      END
```

Appendix G

PLOT Routines Listing

This appendix contains the FORTRAN code for the statistical computations and plotting program (PLOT) used for all filter performance plots in this research. Depending on the variables of interest, the labels can be (and were) changed and appropriate data processed to yield desired plots.

Plot Program

```
PROGRAM PLOTTER     74/74   OPT=2                    FTN 4.7+476        10/

      PROGRAM PLOTTER(OUTPUT,TAPE8,PLOT,TAPE6=OUTPUT)
      REAL KALCORR
      COMMON XM(152),VAR(152),FVAR(152),XMPVAR(152),XMMVAR(152)
      COMMON/MATRIX/KALCORR(14,20,150),STATS(4,3,150),ERROR(2,2,150),
     * ZERO(152),TYME(152),VPT(4,4,21)
      REWIND 8
      NFS=4
      DT=1./30.
      REWIND 8
      NFILE=8
      NTIME=150
      NRUN=20
      CALL PROCESS(NFS,NRUN,NTIME,DT,NFILE)
      CALL OUTPUT(NFS,3,NTIME,8,DT)
      CALL GRAPH(NTIME,NRUN)
      STOP
      END
```

```
SUBROUTINE OUTPUT   74/74   OPT=2                    FTN 4.7+476        10/

      SUBROUTINE OUTPUT(NES,NSTAT,NTIME,NFILE,DT)
      REAL KALCORR
      COMMON/MATRIX/KALCORR(14,20,150),STATS(4,3,150),ERROR(2,2,150),
     * ZERO(152),TYME(152),VPT(4,4,21)
      DO 50 I=1,NES
      TIME=1.
      WRITE(NFILE,1) I
    1 FORMAT(//2X,"**********  THE FOLLOWING IS OUTPUT FOR ERROR STATE ",
     *I1,2X,"**********",//)
      WRITE(NFILE,2)
    2 FORMAT(2X,"TIME",5X,"MEAN ERROR",5X,"STD OF ERROR",5X,"FILTER EST"
     *,"IMATE OF STD"/)
      DO 25 J=1,NTIME
      TIME = TIME+DT
      WRITE(NFILE,3)TIME,(STATS(I,K,J),K=1,NSTAT)
    3 FORMAT(1X,F5.2,3(3X,E12.5))
   25 CONTINUE
   50 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE PROCESS(NES,NRUN,NTIME,DT,NFILE)
      REAL KALCORR
      COMMON/MATRIX/KALCORR(14,20,150),STATS(4,3,150),ERROR(2,2,150),
     * ZERO(152),TYME(152),VPT(4,4,21)
      DIMENSION ESUM(6),SUMSQ(6),RMSUM(4)
      DO 50 I=1,NRUN
      DO 50 J=1,NTIME
      READ(NFILE) (KALCORR(L,I,J),L=1,14)
   50 CONTINUE
      R=FLOAT(NRUN)
      RM = FLOAT(NRUN-1)
      DO 99 I=1,NTIME
      ZERO(I) = 0.
      TYME(I) = FLOAT(I)*DT
      DO 1 K=1,6
      ESUM(K) = 0.
      SUMSQ(K) = 0.
    1 CONTINUE
      DO 2 K=1,4
      RMSUM (K) = 0.
    2 CONTINUE
      DO 3 L=1,4
      DO 3 J=1,NRUN
      ERR = KALCORR(L,J,I) - KALCORR(L+6,J,I)
      ESUM(L) = ESUM(L) + ERR
      SUMSQ(L) = SUMSQ(L) + ERR**2
      RMSUM (L) = RMSUM(L) + KALCORR(L+10,J,I)
    3 CONTINUE
      DO 10 J=1,NRUN
      ERR1 = KALCORR(1,J,I) - KALCORR(9,J,I)
      ERR2 = KALCORR(3,J,I) - KALCORR(10,J,I)
      ESUM(5) = ESUM(5) + ERR1
      ESUM(6) = ESUM(6) + ERR2
      SUMSQ(5) = SUMSQ(5) + ERR1**2
      SUMSQ(6) = SUMSQ(6) + ERR2**2
   10 CONTINUE
      DO 20 K=1,4
      EMEAN = ESUM(K) /R
      ESTDEV = SQRT((SUMSQ(K)-R*EMEAN**2)/RM)
      ESTDEV = SQRT(RMSUM(K)/R)
      STATS(K,1,I) = EMEAN
      STATS(K,2,I) = ESTDEV
      STATS(K,3,I) = ESTDEV
   20 CONTINUE
      DO 21 K=1,2
      EMEAN = ESUM(K+4)/R
      ESTDEV = SQRT((SUMSQ(K+4)-R*EMEAN**2)/RM)
      ERROR(K,1,I) = EMEAN
      ERROR(K,2,I) = ESTDEV
   21 CONTINUE
   99 CONTINUE
      DO 95 L=1,4
      DO 91 I=1,4
      ESUM(I)=KALCORR(L,1,I*30)-KALCORR(L+6,1,I*30)
      SUMSQ(I)=ESUM(I)**2
   91 CONTINUE
      VPT(L,1,1)= VPT(L,2,1)= VPT(L,3,1)= VPT(L,4,1)= 0.
```

```
ROUTINE PROCESS    74/74   OPT=2                          FTN 4.7+476        1
(
            DO 55  I=2,21
            R=FLOAT(I)
            RM=FLOAT(I-1)
            DO 55  J=1,4
            ITIME=J*30
            ERR=KALCORR(L,I,ITIME)-KALCORR(L,I-1,ITIME)
            ESUM(J)= ESUM(J)+ERR
            SUMSQ(J)= SUMSQ(J)+ERR**2
            EMEAN= ESUM(J)/R
            ESTDEV= SQRT((SUMSQ(J)-R*EMEAN**2)/RM)
            VAR(L,J,I)= ESTDEV
         53 CONTINUE
         54 CONTINUE
         55 CONTINUE
            RETURN
            END
```

```
ROUTINE PTNT       74/74   OPT=2                          FTN 4.7+476        1
(
            SUBROUTINE PTNT(A,B,IFLAG)
            REAL KALCORR
            COMMON XM(152),VAR(152),FVAR(152),XMFVAR(152),XMMVAR(152)
            COMMON/MATRIX/KALCORR(14,20,150),STATS(4,3,150),ERROR(2,2,150),
           * ZERO(152),TYME(152),VAR(4,4,21)
            DIMENSION A(17),B(17)
            VAR(151)=FVAR(151)=XMFVAR(151)=XMMVAR(151)=0.
            VAR(152)=FVAR(152)=XMFVAR(152)=XMMVAR(152)=0.
            CALL SCALE(XMFVAR,4.5,302,1)
            XMFVAR(151)=XM(151)=ZERO(151)=XMMVAR(151)
            XMFVAR(152)=XM(152)=ZERO(152)=XMMVAR(152)=-XMMVAR(152)
            CALL VGRAPH(TYME,XM,150,A,-1,15,4)
            CALL VGRAPH(TYME,XMFVAR,150,A,2,15,14)
            CALL VGRAPH(TYME,XMMVAR,150,A,2,15,14)
            IF(IFLAG.EQ.0)GO TO 99
            CALL SCALE(VAR,4.5,302,1)
            VAR(151)=FVAR(151)
            VAR(152)=FVAR(152)=-FVAR(152)
            CALL VGRAPH(TYME,VAR,150,B,-2,15,4)
            CALL VGRAPH(TYME,FVAR,150,B,2,15,4)
         99 RETURN
            END
```

```
      SUBROUTINE GRAPH(NTIME,NRUN)
      REAL KALCORR
      COMMON XM(152),VAR(152),FVAR(152),XMFVAR(152),XMKVAR(152)
      COMMON/MATRIX/KALCORR(14,20,150),STATS(4,3,150),ERROR(2,2,150),
     * ZERO(152),TYPE(152),VPT(4,4,21)
      DIMENSION AD(17),BD(17),CD(17),DD(17),FD(17),ED(17),GD(17),HD(17)
     *,OD(17),RD(17),SD(17),TD(17)
      DATA AD(1)/2HMEAN ERROR +-1 SIGMA/
      DATA AD(3)/2H  TARGET DYNAMICS   /
      DATA AD(5)/2H      X CHANNEL     /
      DATA AD(7)/2H       FLTR FOV     /
      DATA AD(9)/2H      TIME (SEC)    /
      DATA AD(11)/2HH        PIXELS     /
      DATA AD(13)/2HH           X CHANNEL DYNAMICS ERROR (S/N=   )      /
      BD(1)=CD(1)=DD(1)=AD(1)
      BD(2)=CD(2)=DD(2)=AD(2)
      BD(7)=CD(7)=DD(7)=ED(7)=FD(7)=GD(7)=HD(7)=AD(7)
      BD(8)=CD(8)=DD(8)=ED(8)=FD(8)=GD(8)=HD(8)=AD(8)
      BD(9)=CD(9)=DD(9)=ED(9)=FD(9)=GD(9)=HD(9)=OD(9)=RD(9)=SD(9)=TD(9)=
     *AD(9)
      BD(10)=CD(10)=DD(10)=ED(10)=FD(10)=GD(10)=HD(10)=OD(10)=RD(10)=
     *SD(10)=TD(10)=AD(10)
      BD(11)=CD(11)=DD(11)=ED(11)=FD(11)=GD(11)=HD(11)=OD(11)=RD(11)=
     *SD(11)=TD(11)=AD(11)
      BD(12)=CD(12)=DD(12)=ED(12)=FD(12)=GD(12)=HD(12)=OD(12)=RD(12)=
     *SD(12)=TD(12)=AD(12)
      BD(5)=ED(5)=GD(5)=OD(5)=RD(5)=AD(5)
      BD(6)=ED(6)=GD(6)=OD(6)=RD(6)=AD(6)
      DATA CD(5)/2H      Y CHANNEL     /
      DD(5)=FD(5)=HD(5)=SD(5)=TD(5)=CD(5)
      DD(6)=FD(6)=HD(6)=SD(6)=TD(6)=CD(6)
      DATA BD(3)/2H      VELOCITY      /
      DD(3)=RD(7)=TD(7)=BD(3)
      DD(4)=FD(8)=TD(8)=BD(4)
      CD(3)=OD(7)=SD(7)=AD(7)
      CD(4)=CD(8)=SD(8)=AD(4)
      DATA ED(13)/2HH           X CHANNEL VELOCITY ERROR (S/N=   )      /
      DATA OD(1)/2H ACTUAL VS. FILTER  /
      DATA CD(3)/2H       SIGMA        /
      RD(1) = SD(1) = TD(1) = OD(1)
      RD(2) = SD(2) = TD(2) = OD(2)
      RD(3) = SD(3) = TD(3) = OD(3)
      RD(4) = SD(4) = TD(4) = OD(4)
      DATA CD(13)/2HH           Y CHANNEL DYNAMICS ERROR (S/N=  )       /
      DATA DD(13)/2HH           Y CHANNEL VELOCITY ERROR (S/N=   )      /
      DATA CD(13)/2HH           FILTER VS. ACTUAL SIGMA PLOT  (S/N =  )/
      DATA ED(1)/2HMEAN TRACKING ERROR /
      DATA ED(3)/2H  FOR FILTER MODEL  /
      SD(13)=TD(13)=RD(13)=OD(13)
      SD(14)=TD(14)=RD(14)=OD(14)
      SD(15)=TD(15)=RD(15)=OD(15)
      SD(16)=TD(16)=RD(16)=OD(16)
      SD(17)=TD(17)=RD(17)=OD(17)
      DATA GD(3)/2HFOR CORRELATOR MODEL/
      FD(1) = GD(1) = HD(1) = ED(1)
      FD(2) = GD(2) = HD(2) = ED(2)
      FD(3)=ED(3)
```

```
SUBROUTINE GRAPH      74/74   OPT=2                    FTN 4.7+476        1:
      ED(4)=ED(4)
      HD(3)=GD(3)
      HD(4)=GD(4)
      DATA ED(13)/5HX CHANNEL FILTER TRACKING ERROR (S/N =  )/
      DATA FD(13)/5HY CHANNEL FILTER TRACKING ERROR (S/N =  )/
      DATA GD(13)/5HX CHANNEL CORRELATOR TRACKING ERROR (S/N =  )/
      DATA HD(13)/5HY CHANNEL CORRELATOR TRACKING ERROR (S/N =  )/
      CALL PLOTS(3HA,0,3HPLOT)
      CALL PLOT(1.63,-2.1,-3)
      CALL SCALE(TIME,7.,150,1)
      DO 2  M=1,4
      DO 1 I=1,NTIME
      XM(I) = STATS(M,1,I)
      XMPVAR(I) = STATS(M,1,I) + STATS(M,2,I)
      XMNVAR(I) = STATS(M,1,I) - STATS(M,2,I)
      FVAR(I) = STATS(M,3,I)
      VAR(I) = STATS(M,2,I)
    1 CONTINUE
      CALL PLOT(1.75,-2.1,-3)
      CALL DSP(2HRR)
      GO TO (6,7,8,9),M
    6 CALL PTNT(AD,ID,1)
      CALL VARPT(AD,NRUN,1)
      GO TO 20
    7 CALL PTNT(BD,ID,1)
      CALL VARPT(BD,NRUN,2)
      GO TO 20
    8 CALL PTNT(CD,ID,1)
      CALL VARPT(CD,NRUN,3)
      GO TO 20
    9 CALL PTNT(DD,ID,1)
      CALL VARPT(DD,NRUN,4)
   20 CONTINUE
      CALL PLOT(1.75,-2.1,-3)
      CALL DSP(2HRB)
      CALL PLOTE(R)
      RETURN
      END
```

```
      SUBROUTINE VARPT(TD,N,I)
      REAL KALCORR
      COMMON XX(152),VAR(152),FVAR(152),XMFVAR(152),XMNVAR(152)
      COMMON/MATRIX/KALCORR(14,20,150),STATS(4,3,150),ERROR(2,2,150),
     * ZERO(152),TYMF(152),VPT(4,4,21)
      DIMENSION Y(22),ID(17),RUNS(22),DUM(84),VPLOT(4,21)
      EQUIVALENCE(DUM(1),VPLOT(1,1))
      DO 10 I=1,4
      DO 10 J=1,21
      VPLOT(I,J)= VPT(L,I,J)
   10 CONTINUE
      CALL PLOT(1.85,-2.1,-3)
      CALL PLOT(3.5,.1,2)
      CALL PLOT(3.5,10.5,3)
      CALL PLOT(8.5,11.,2)
      CALL PLOT(3.5,11.,3)
      CALL PLOT(3.5,10.5,2)
      CALL PLOT(8.5,.1,3)
      CALL PLOT(3.5,.1,2)
      CALL PLOT(3.5,.1,3)
      CALL PLOT(1.35,1.35,3)
      CALL PLOT(1.35,9.65,2)
      CALL PLOT(1.65,9.65,3)
      CALL PLOT(1.65,7.65,2)
      ID(1)=19H        JAPT
      ID(2)=19HANCE
      DO 20 I=1,7,2
      CALL SYMBOL((1.65+(I+1.5)*.1),7.35,1.10,ID(I),90.,20)
   20 CONTINUE
      ID(1)=20H      MEAN ERROR
      CALL PLOT(1.65,7.65,3)
      CALL PLOT(2.65,7.65,2)
      CALL PLOT(2.65,9.65,2)
      CALL PLOT(1.65,9.65,2)
      CALL PLOT(1.35,9.65,3)
      CALL PLOT(7.65,9.65,2)
      CALL PLOT(7.65,1.35,2)
      CALL PLOT(1.35,1.35,2)
      CALL SYMBOL(1.65,1.115,.105,
     XH                          VARIANCE CONVERGENCE          ,0.,50)
      CALL PLOT(8.95,2.1,-3)
      CALL SCALE(DUM,6.5,21,1)
      DO 30 I=1,N
      RUNS(I)=FLOAT(I)
   30 CONTINUE
      RUNS(N+1)=1.RUNS(N+2)=7
      CALL AXIS(0.,0.,8HVARIANCE,8,6.5,180.,DUM(81),DUM(82))
      CALL AXIS(0.,0.,3HRUN,-3,7.,90.,1.,3.)
      Y(N+1)= DUM(81)
      Y(N+2)=-DUM(82)
      DO 40 J=1,4
      DO 50 I=1,N
      Y(I)= VPLOT(J,I)
   50 CONTINUE
      CALL LINE(Y,RUNS,N,1,1,J)
   40 CONTINUE
      RETURN
```

```
ROUTINE VGRAPH     74/74   OPT=2                                FTN 4.7+476      12
      SUBROUTINE VGRAPH(X,Y,N,ID,NO,NP,NS)
      REAL KALCORR
      COMMON XM(152),VAR(152),FVAR(152),X1FVAR(152),XMMVAR(152)
      COMMON/MATRIX/KALCORR(14,20,150),STATS(4,3,150),ERROR(2,2,150),
     * ZERO(152),TYME(152),VFT(4,4,21)
      DIMENSION X(152),Y(152),ID(17)
      IF(NO.EQ.2) GO TO 30
      CALL PLOT(1.55,-2.1,-3)
      CALL PLOT(0.,.1,2)
      CALL PLOT(0.,10.9,3)
      CALL PLOT(0.,11.,2)
      CALL PLOT(8.5,11.,3)
      CALL PLOT(8.5,10.9,2)
      CALL PLOT(8.5,.1,3)
      CALL PLOT(8.5,0.,2)
      CALL PLOT(0.,0.,3)
      CALL PLOT(1.35,1.35,3)
      CALL PLOT(1.35,9.65,2)
      CALL PLOT(1.45,9.55,3)
      CALL PLOT(1.45,7.55,2)
      DO 20 I=1,7,2
      CALL SYMBOL((1.45+(I+1.5)*.1),7.35,.03,ID(I),90.,20)
   20 CONTINUE
      CALL PLOT(1.45,7.55,3)
      CALL PLOT(2.45,7.55,2)
      CALL PLOT(2.45,9.55,2)
      CALL PLOT(1.45,9.55,2)
      CALL PLOT(1.35,9.65,3)
      CALL PLOT(7.45,9.65,2)
      CALL PLOT(7.45,1.35,2)
      CALL PLOT(1.35,1.35,2)
      CALL SYMBOL(1.55,1.115,.105,ID(13),0.,50)
      CALL PLOT(5.95,2.1,-3)
      Y2=-Y(N+2)
      Y1= Y(N+1)
      X1= X(N+1)
      X2= X(N+2)
      CALL AXIS(0.,0.,ID(9),-20,7.,90.,X1,X2)
      CALL AXIS(0.,0.,ID(11),20,4.5,180.,Y1,Y2)
      IF(NO.EQ.-2)GO TO 31
      CALL LINE(ZERO,TYME,N,1,0,NS)
   30 CONTINUE
      CALL LINE(Y,X,N,1,NP,NS)
      RETURN
      END
```

Appendix H

Parameters for Cases 1-31

Symbol	Case / Fortran Code	Refer to List of Symbols for Description of Variables									
		1	2	3	4	5	6	7	8	9	10
τ_D^2	DTAU	1	1	.2	.5	1	1	1	1	1	1
σ_D^2	SIGS1	1	1	1	1	1	1	1	1	1	1
σ_N^2	SIGMAB	1	1	1	1	1	1	1	1	10	.5
σ_f	SIGFLR	0	0	0	0	0	0	0	0	0	0
I_{max}	IMAX	20	10	10	10	1	20	10	10	10	10
SN	SN	20	10	10	10	1	20	10	10	1	20
σ_D/σ_A	SDSA	5	1	1	1	1	1	1	1	1	1
-	NRUN	20	20	20	20	20	20	20	20	20	20
t_f	TFINAL	5	5	5	5	5	5	5	5	5	5
σ_A	SIGF2	.2	1	1	1	1	1	1	1	1	1
σ_g	SIGMS	3	3	3	3	3	3	1	5	3	3
σ_{gF}	SIGMF	3	3	3	3	3	3	3	3	3	3
-	ICORR	0	0	0	0	0	0	0	0	0	0
σ_R	RF	1	1	1	1	1	1	1	1	1	1
σ_{DF}	SIGF1	1	1	1	1	1	1	1	1	1	1
τ_{DF}	FTAU1	1	1	1	1	1	1	1	1	1	1
I	FIMAX	20	10	10	10	10	10	10	10	10	10
AR	ASPRO	-	-	-	-	-	-	-	-	-	-
-	VXO	-	-	-	-	-	-	-	-	-	-
-	VYO	-	-	-	-	-	-	-	-	-	-
τ_N	DTIM	-	-	-	-	-	-	-	-	-	-
-	ISPTL	-	-	-	-	-	-	-	-	-	-

Symbol	Case / Fortran Code	Refer to List of Symbols for Description of Variables										
		1	2	3	4	5	6		7	8	9	10
$r_{k,k+1}$	C(1)	-	-	-	-	-	-		-	-	-	-
$r_{k,k+9}$	C(2)	-	-	-	-	-	-		-	-	-	-
$r_{k,k+2}$	C(3)	-	-	-	-	-	-		-	-	-	-
$r_{k,k+10}$	C(4)	-	-	-	-	-	-		-	-	-	-
$r_{k,k+18}$	C(5)	-	-	-	-	-	-		-	-	-	-

Symbol	Fortran Code / Case	Refer to List of Symbols for Description of Variables										
		11	12		13	14	15	16	17	18	19	20
τ_D^2	DTAU	1	1		1	1	1	1	1	1	1	1
σ_D^2	SIGS1	.2	5		1	1	1	1	1	1	1	1
σ_N^2	SIGMAB	1	1		1	1	1	1	1	1	1	1
σ_f	SIGFLR	0	0		0	0	0	0	0	0	0	0
I_{max}	IMAX	10	10		10	10	10	10	10	1	10	10
SN	SN	10	10		10	10	10	10	10	1	10	10
σ_D/σ_A	SDSA	.2	5		1	1	1	1	1	1	5	5
-	NRUN	20	20		20	20	20	20	20	20	20	20
t_f	TFINAL	5	5		5	5	5	5	5	5	5	5
σ_A	SIGF2	1	1		1	1	1	1	1	1	.2	.2
σ_g	SIGMS	3	3		1	1	1	3	3	3	3	3
σ_{gF}	SIGMF	3	3		1	1	1	3	3	3	3	3
-	ICORR	0	0		0	0	0	0	0	0	0	0
σ_R	RF	1	1		1	1	1	1	1	1	1	1
σ_{DF}	SIGF1	1	1		1	1	1	1	1	1	1	1
τ_{DF}	FTAU1	1	1		1	1	1	1	1	1	1	1
I	FIMAX	10	10		10	10	10	10	10	1	10	10
AR	ASPRO	1	1		2	5	10	1	1	1	1	1
-	VXO	-	-		-	-	-	-	-	-	0	20
-	VYO	-	-		-	-	-	-	-	-	0	20
τ_N	DTIM	-	-		-	-	-	-	-	-	.01	.01
-	ISPTL	-	-		-	-	-	1	1	1	0	0

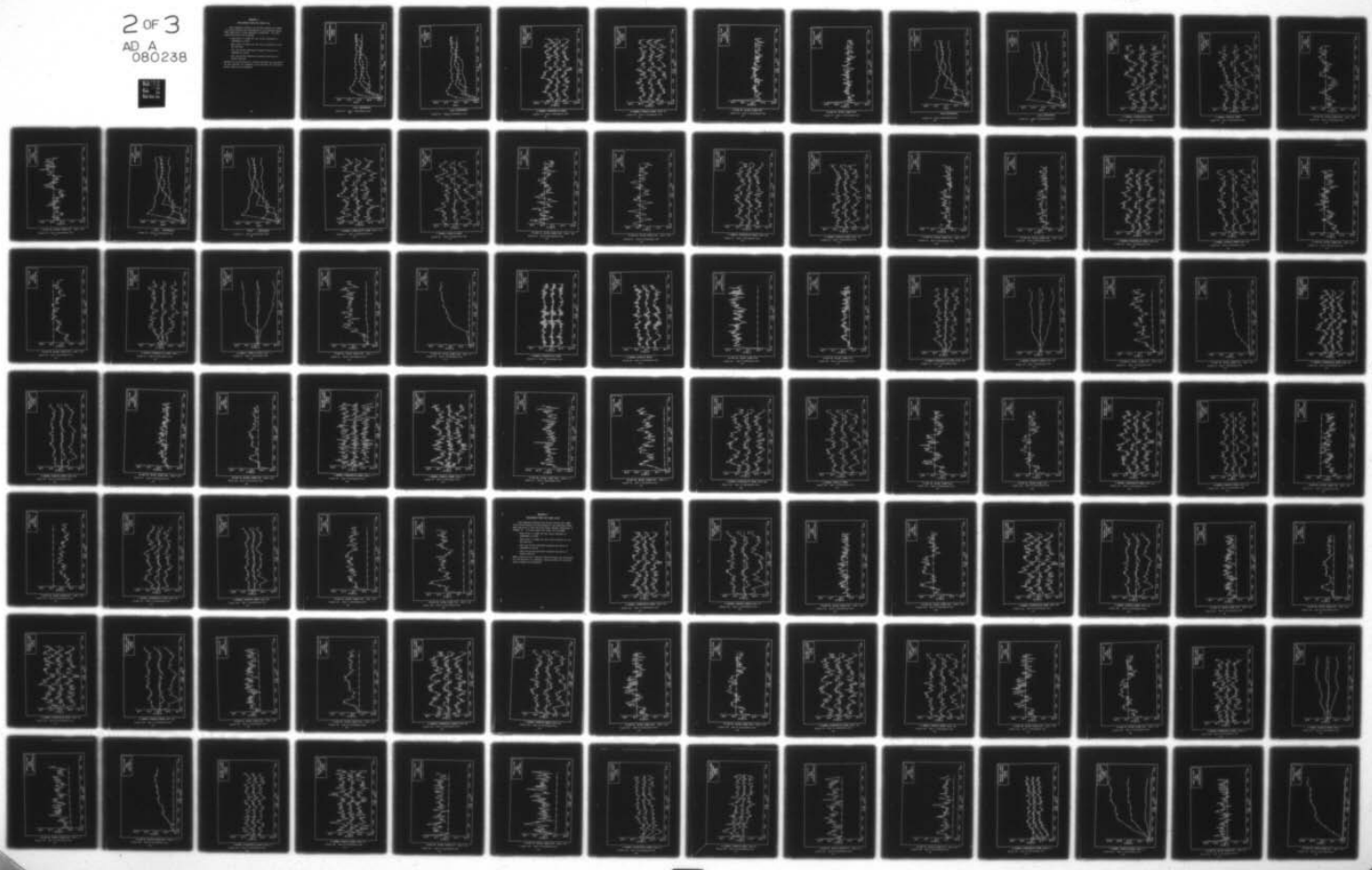
Symbol	Case / Fortran Code	Refer to List of Symbols for Description of Variables										
		11	12		13	14	15	16	17	18	19	20
$r_{k,k+1}$	C(1)	-	-		-	-	-	.368	.497	.497	.0	0
$r_{k,k+9}$	C(2)	-	-		-	-	-	.243	.372	.372	.0	0
$r_{k,k+2}$	C(3)	-	-		-	-	-	.135	.247	.247	.0	0
$r_{k,k+10}$	C(4)	-	-		-	-	-	.107	.209	.209	.0	0
$r_{k,k+18}$	C(5)	-	-		-	-	-	.059	.138	.138	.0	0

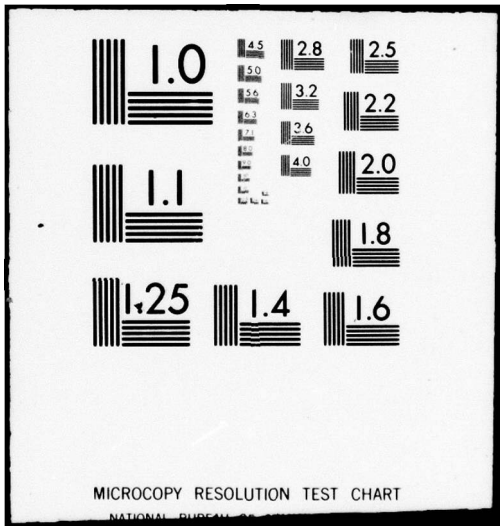
Symbol	Fortran Code / Case	Refer to List of Symbols for Description of Variables										
		21	22	23	24	25	26	27	28	29	30	31
τ_D^2	DTAU	1	1	1	1	1	1	1	1	1	1	1
σ_D^2	SIGS1	1	1	1	1	1	1	1	1	1	1	1
σ_N^2	SIGMAB	1	1	1	1	1	1	1	1	1	1	1
σ_f	SIGFLR	0	0	0	0	0	0	0	0	0	0	0
I_{max}	IMAX	10	10	10	1	2	2	2	2	2	2	2
SN	SN	10	10	10	1	2	2	2	2	2	2	2
σ_D/σ_A	SDSA	5	5	5	5	5	5	5	5	5	5	5
-	NRUN	20	20	20	20	20	20	20	20	20	20	20
t_f	TFINAL	5	5	5	5	5	5	5	5	5	5	5
σ_A	SIGF2	.2	.2	.2	.2	.2	.2	.2	.2	.2	.2	.2
σ_g	SIGMS	3	3	3	3	3	3	3	3	3	3	3
σ_{gF}	SIGMF	3	3	3	3	3	3	3	3	3	3	3
-	ICORR	0	0	0	0	0	0	0	0	0	0	0
σ_R	RF	1	1	1	1	1	1	1	1	1	1	1
σ_{DF}	SIGF1	1	1	1	1	1	1	1	1	1	1	1
τ_{DF}	FTAU1	1	1	1	1	1	1	1	1	1	1	1
I	FIMAX	10	10	10	1	2	2	2	2	2	2	2
AR	ASPRO	1	1	1	1	1	1	1	1	1	1	1
-	VXO	60	40	0	0	0	0	0	0	0	20	20
-	VYO	60	40	0	0	0	0	0	0	0	20	20
τ_N	DTIM	.01	.01	20	20	.01	20	1.4	.01	20	.01	20
-	ISPTL	0	0	1	1	0	1	1	1	0	0	1

Symbol	Case / Fortran Code	Refer to List of Symbols for Description of Variables										
		21	22	23	24	25	26	27	28	29	30	31
$r_{k,k+1}$	C(1)	0	0	.368	.368	0	.368	.368	.368	0	0	.368
$r_{k,k+9}$	C(2)	0	0	.243	.243	0	.243	.243	.243	0	0	.243
$r_{k,k+2}$	C(3)	0	0	.135	.135	0	.135	.135	.135	0	0	.135
$r_{k,k+10}$	C(4)	0	0	.107	.107	0	.107	.107	.107	0	0	.107
$r_{k,k+18}$	C(5)	0	0	.059	.059	0	.059	.059	.059	0	0	.059

AD-A080 238 AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH SCHOO--ETC F/G 17/5
AN ADAPTIVE DISTRIBUTED-MEASUREMENT EXTENDED KALMAN FILTER FOR --ETC(U)
DEC 79 R L JENSEN, D A HARNLY
UNCLASSIFIED AFIT/GA/EE/79-1-VOL-2 NL

2 OF 3
AD A
080238




1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0
4.5
5.0
5.6
6.3
7.1
MICROCOPY RESOLUTION TEST CHART

Appendix I

Performance Plots for Cases 1-12

This appendix contains the plotted outputs for cases 1-12 which studied the performance of the previous filter when subjected to input parameter mismatches. For most cases four plots are included:

- mean error ±1 sigma for the filter estimate of atmospheric jitter
- mean error ±1 sigma for the filter estimate of target position
- real and filter-indicated standard deviation of atmospheric jitter
- real and filter-indicated standard deviation of target position

Because of the similarity in results between the horizontal (x) and vertical (y) channels, only the plots for the horizontal channel are presented.

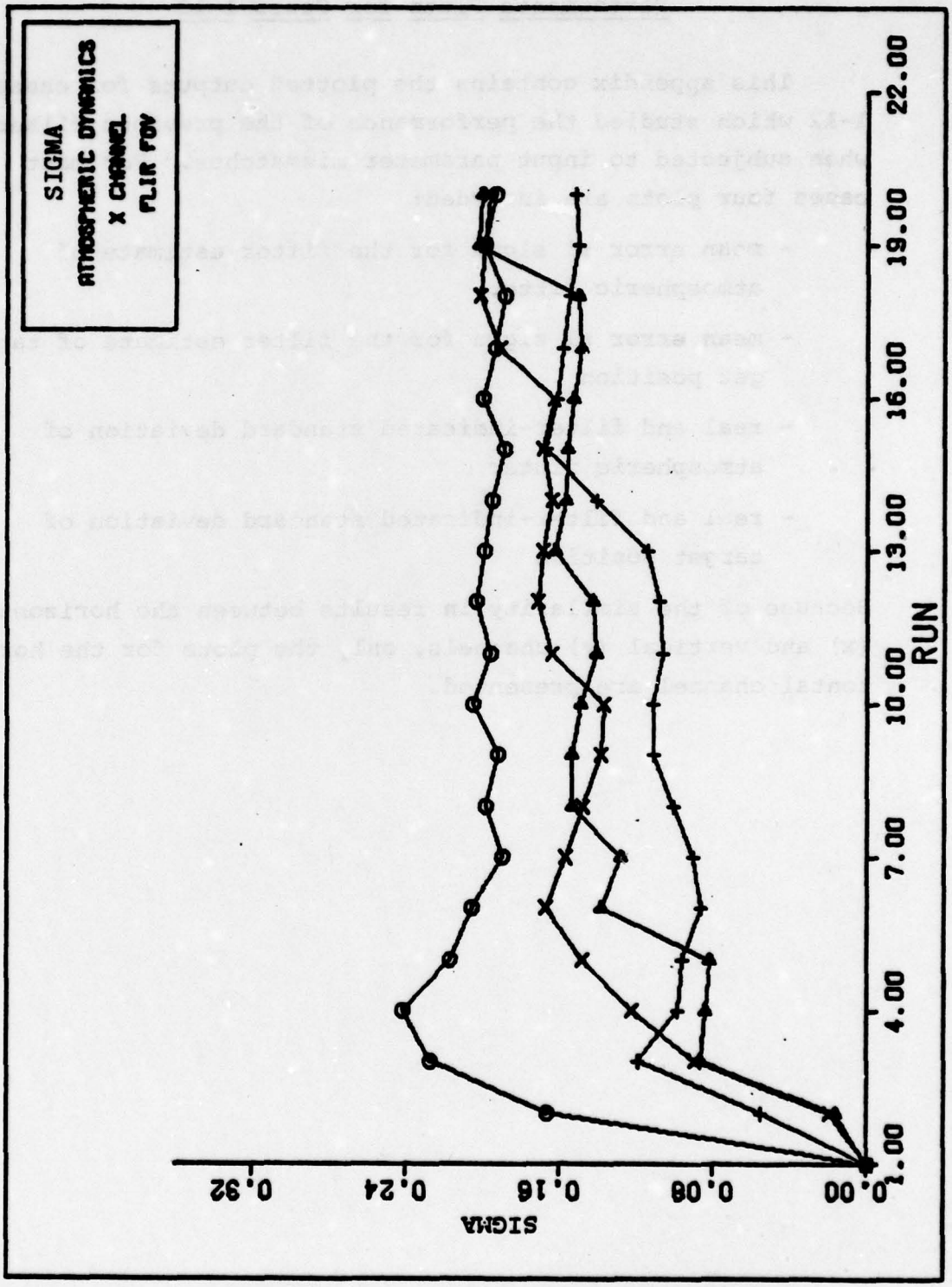
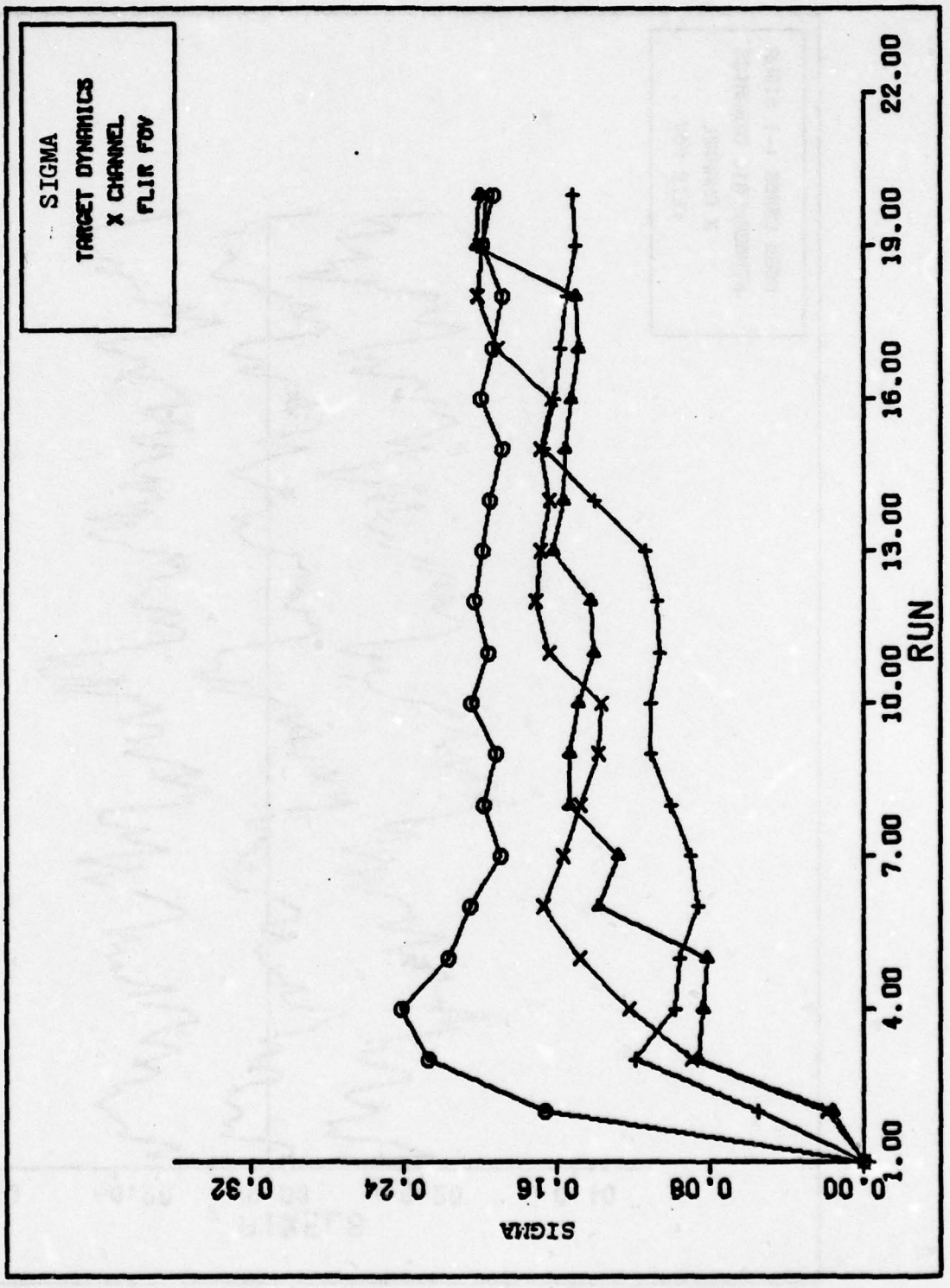
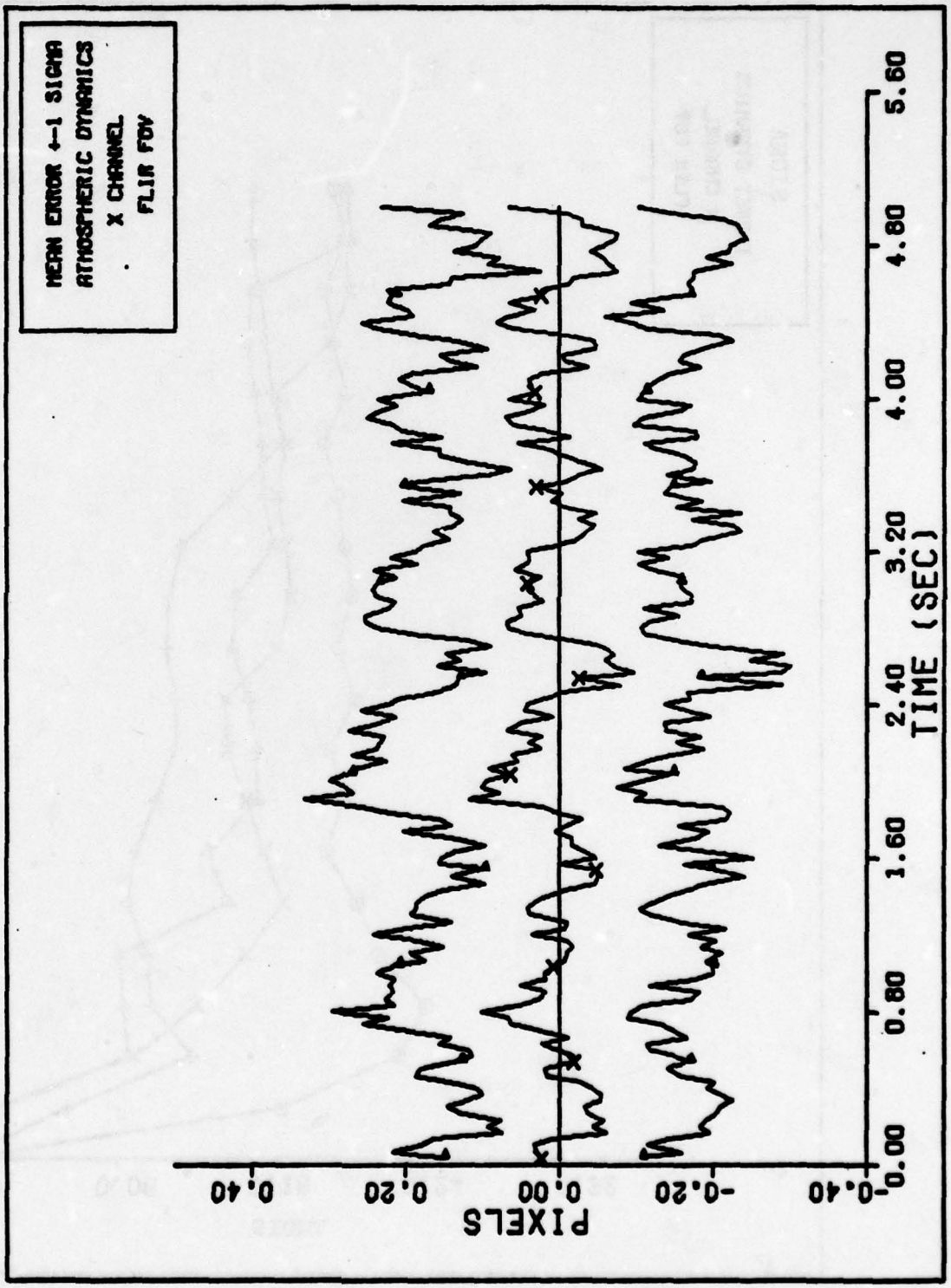


Figure 1a. Case 1 Performance Plot



SIGMA CONVERGENCE

Figure 1b. Case 1 Performance Plot



X CHANNEL ATMOSPHERICS ERROR

Figure 1c. Case 1 Performance Plot

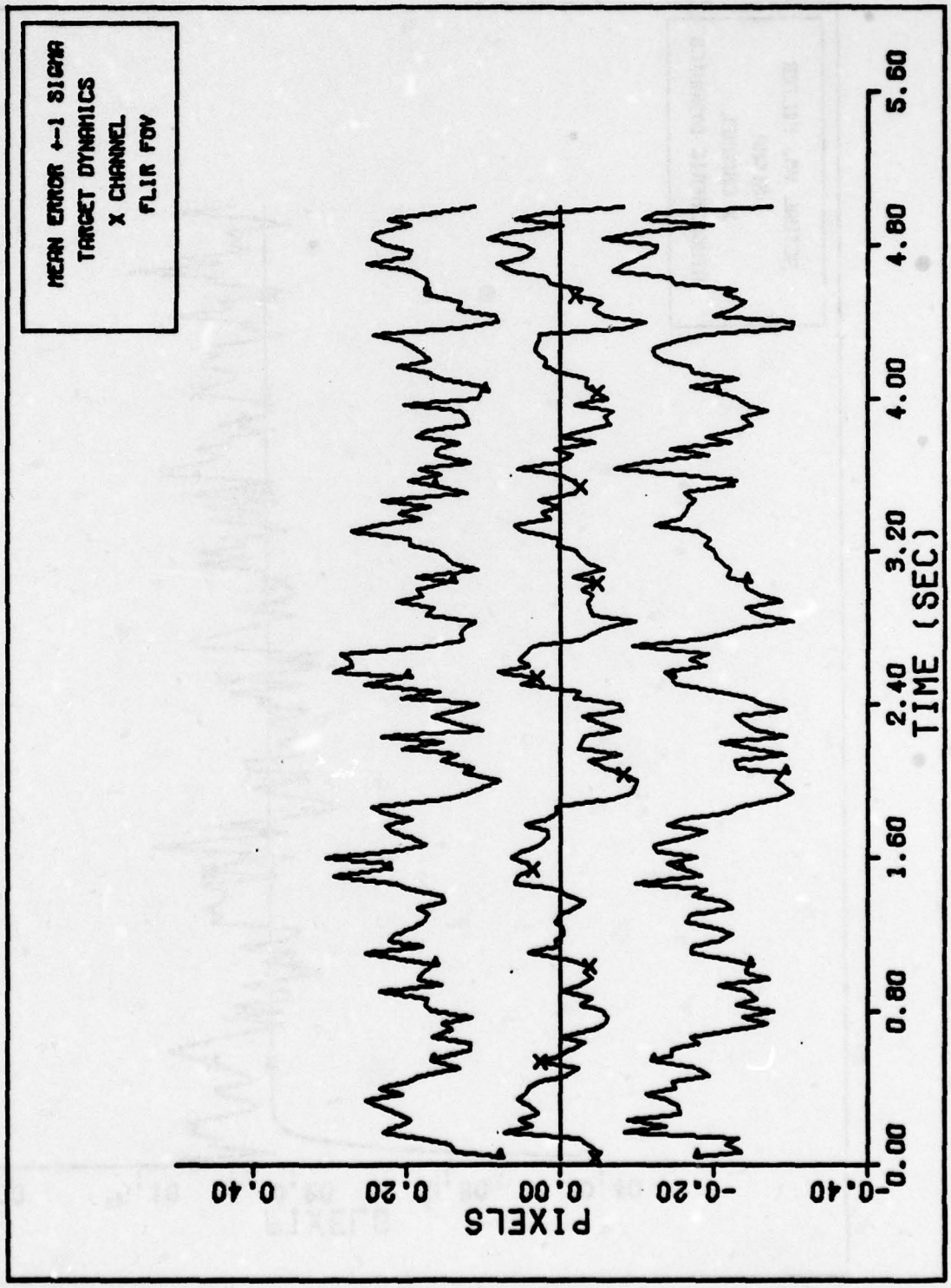
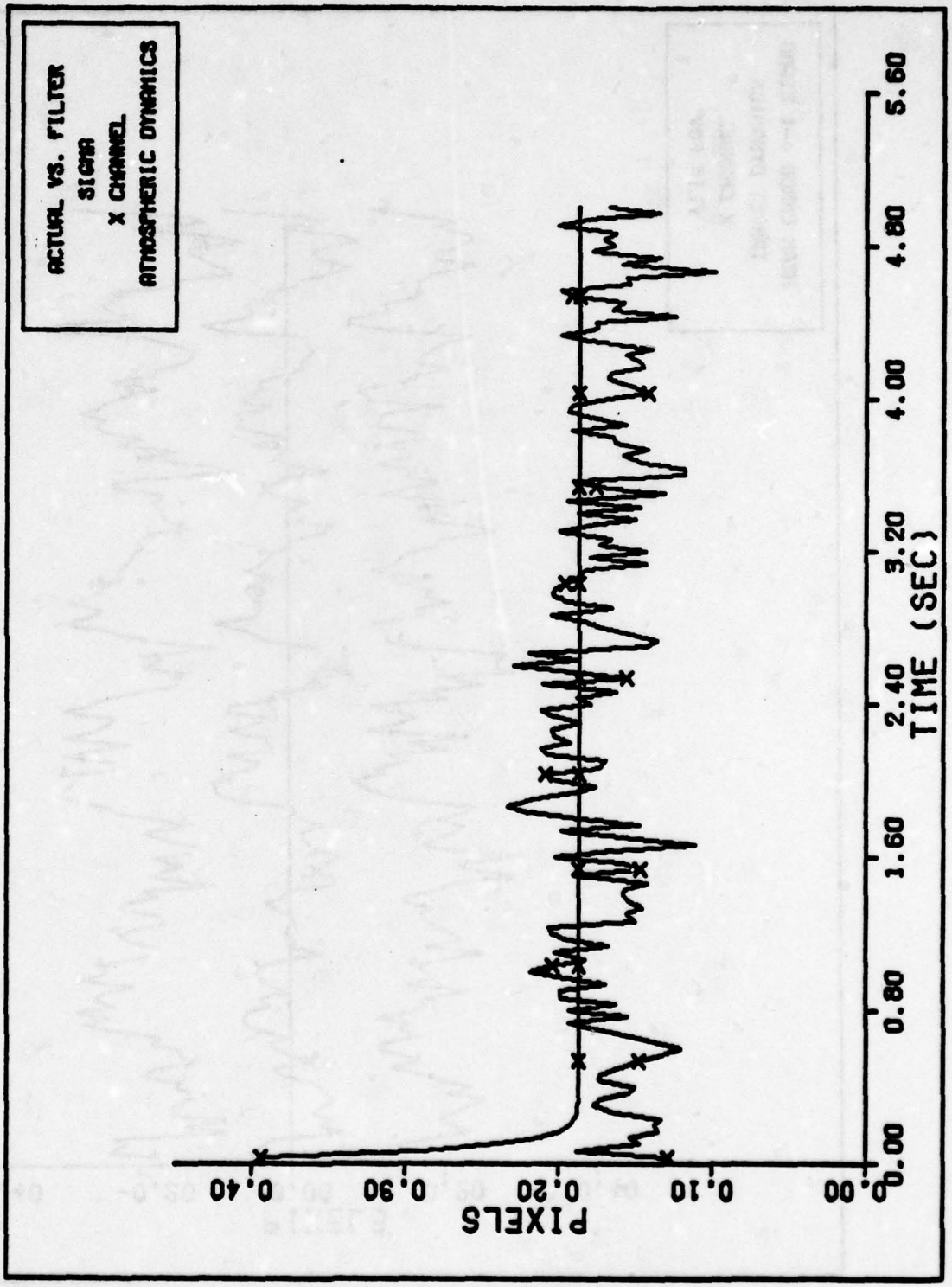
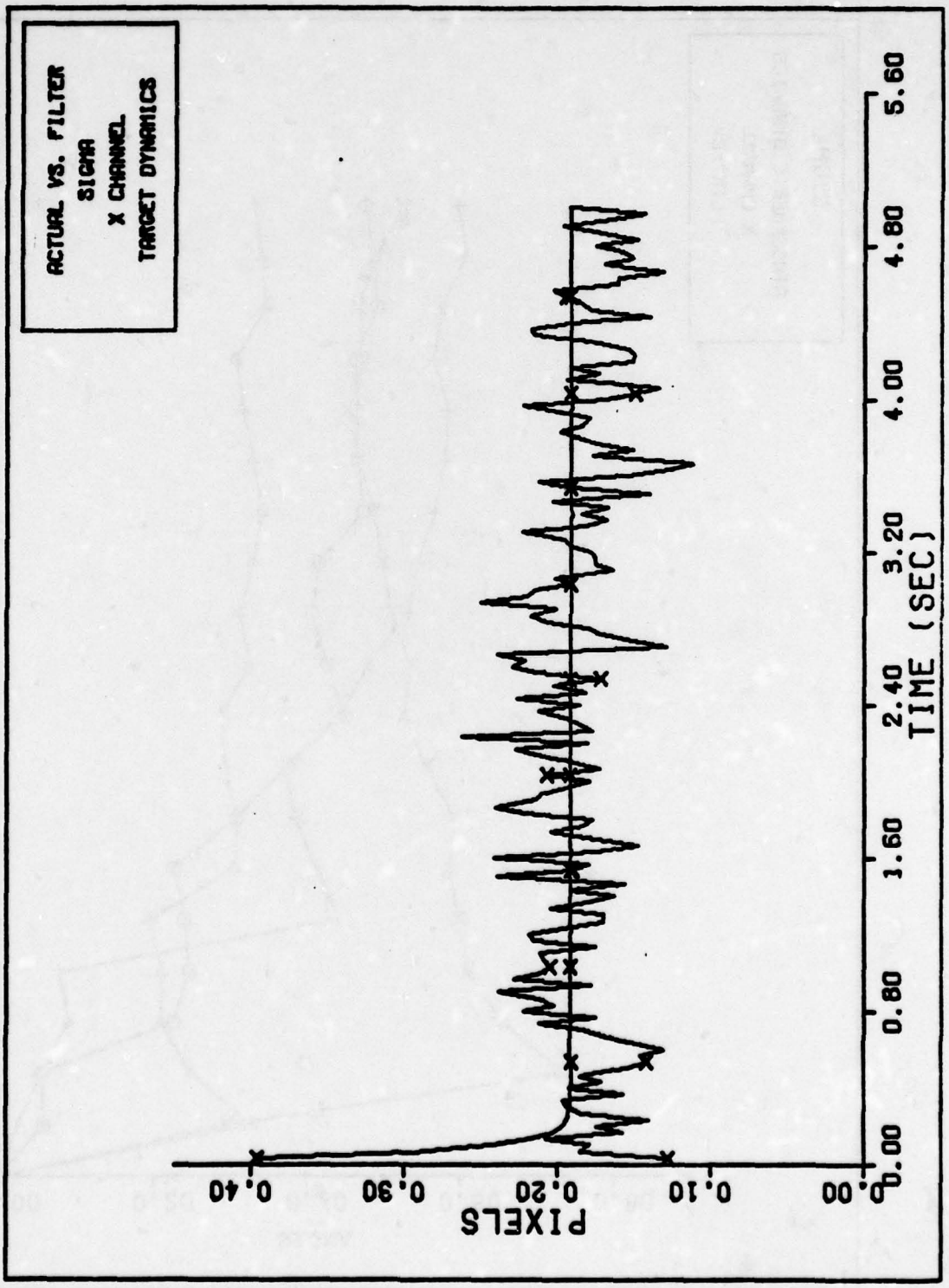


Figure 1d. Case 1 Performance Plot



FILTER VS. ACTUAL SIGMA PLOT

Figure 1e. Case 1 Performance Plot



FILTER VS. ACTUAL SIGMA PLOT

Figure 1f. Case 1 Performance Plot

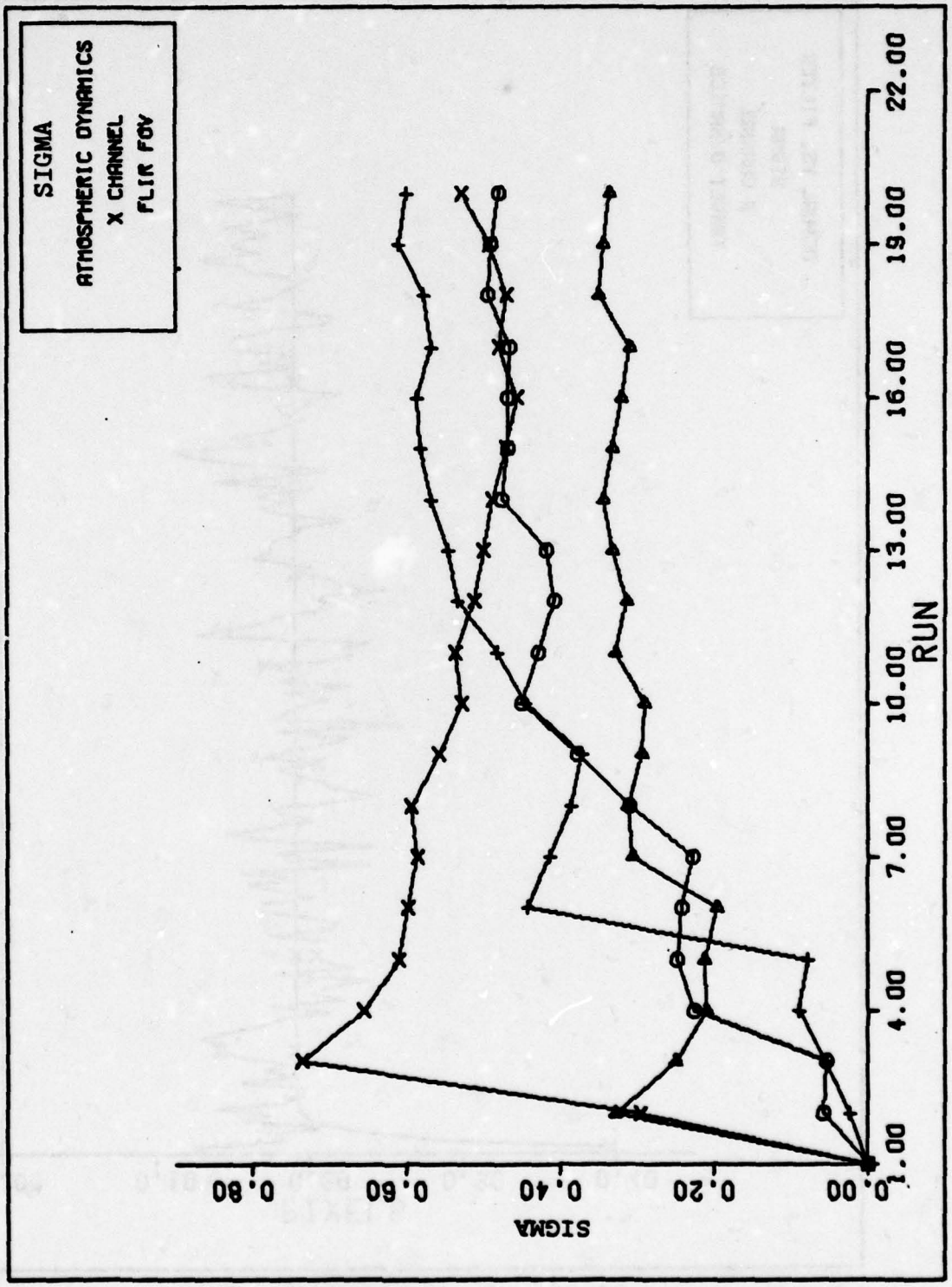


Figure 2a. Case 2 Performance Plot

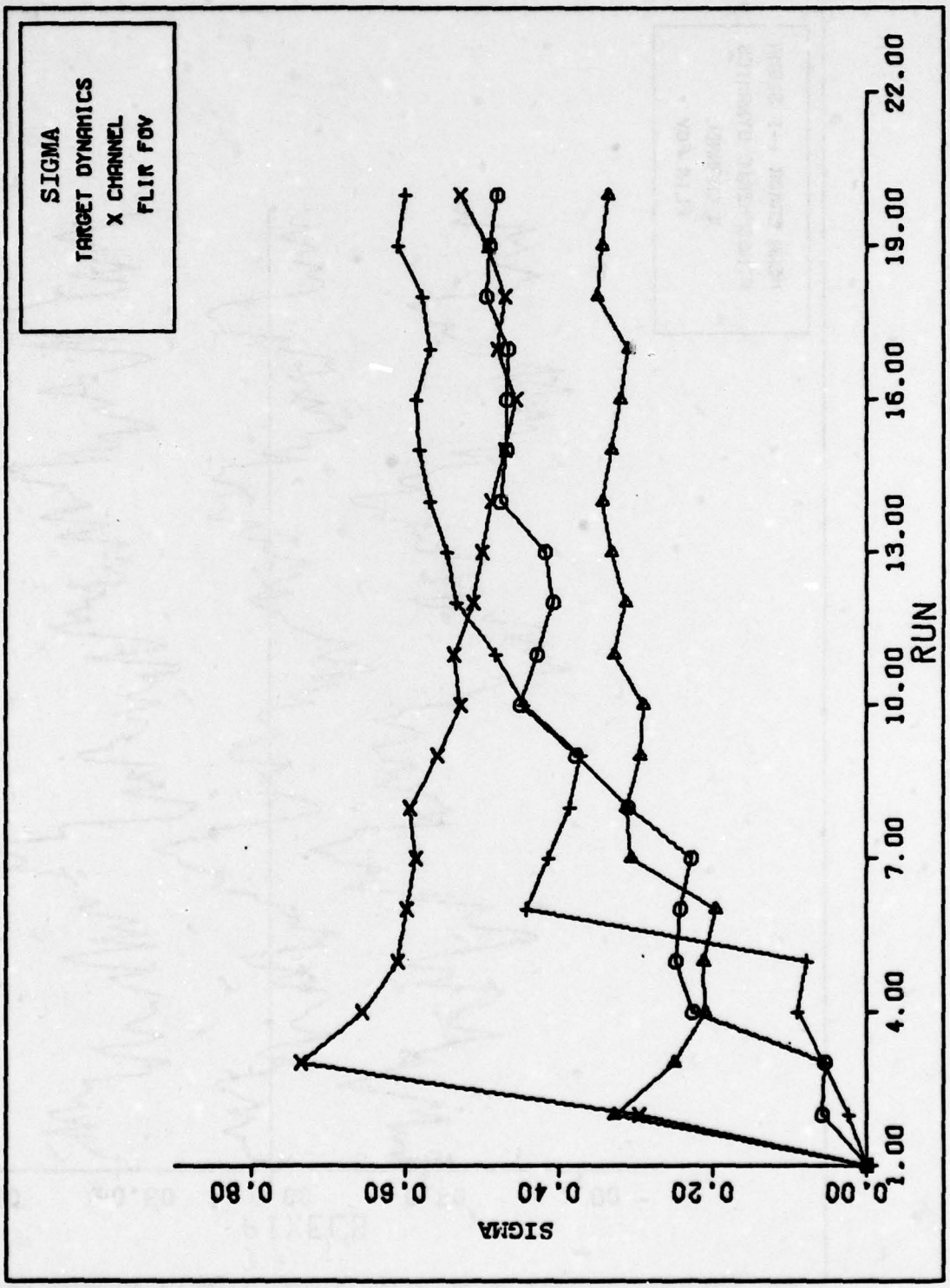
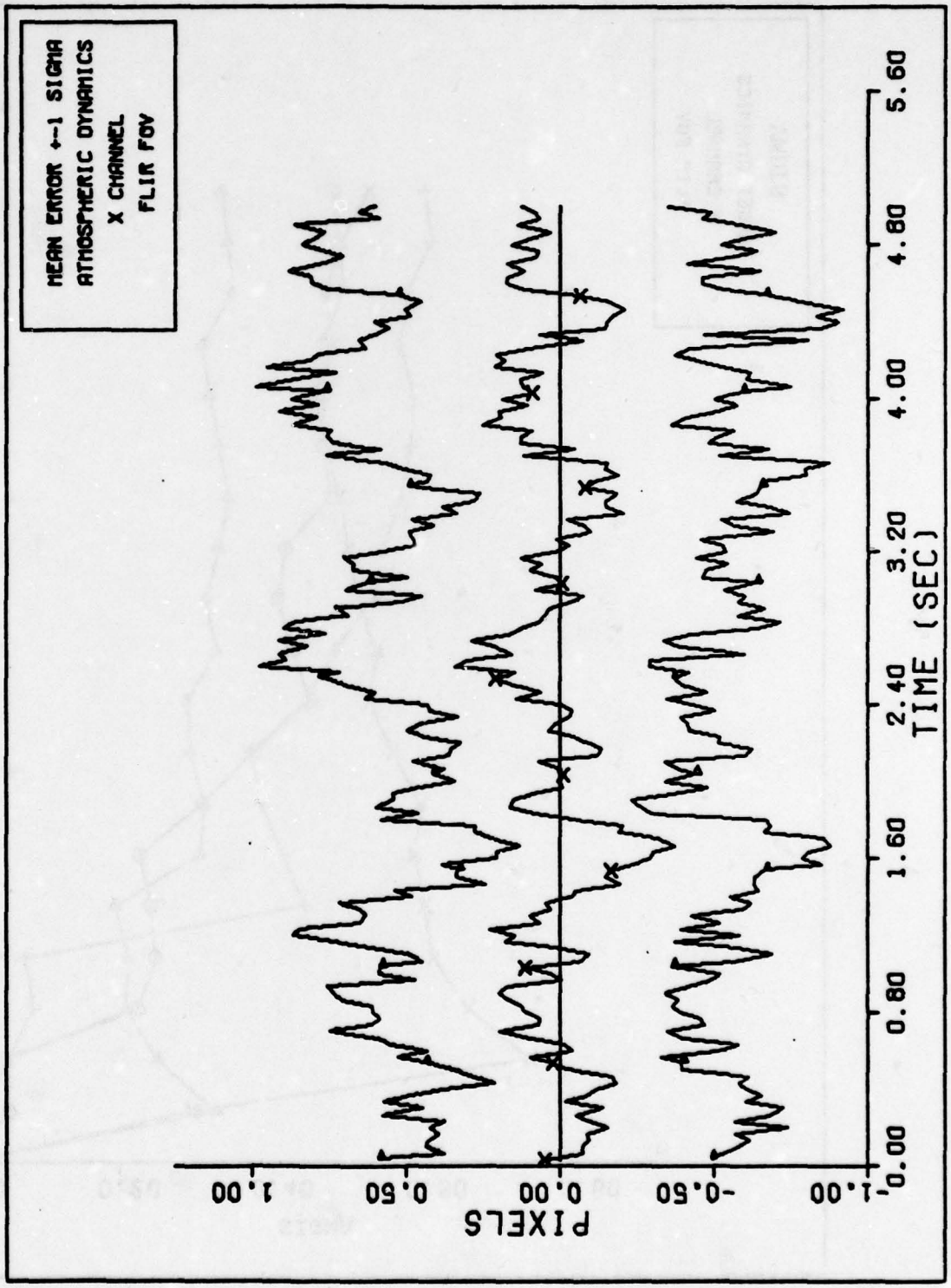
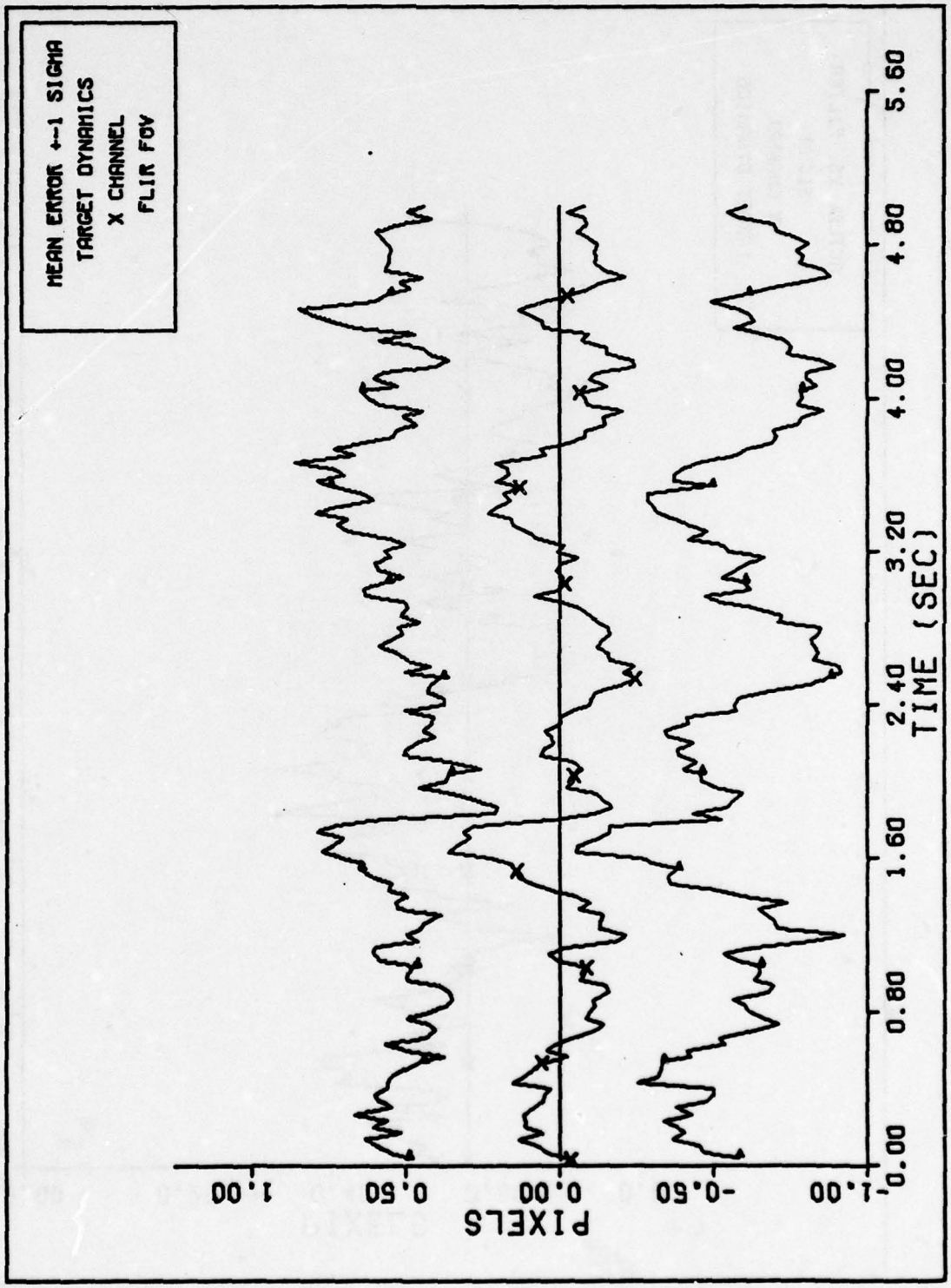


Figure 2b. Case 2 Performance Plot



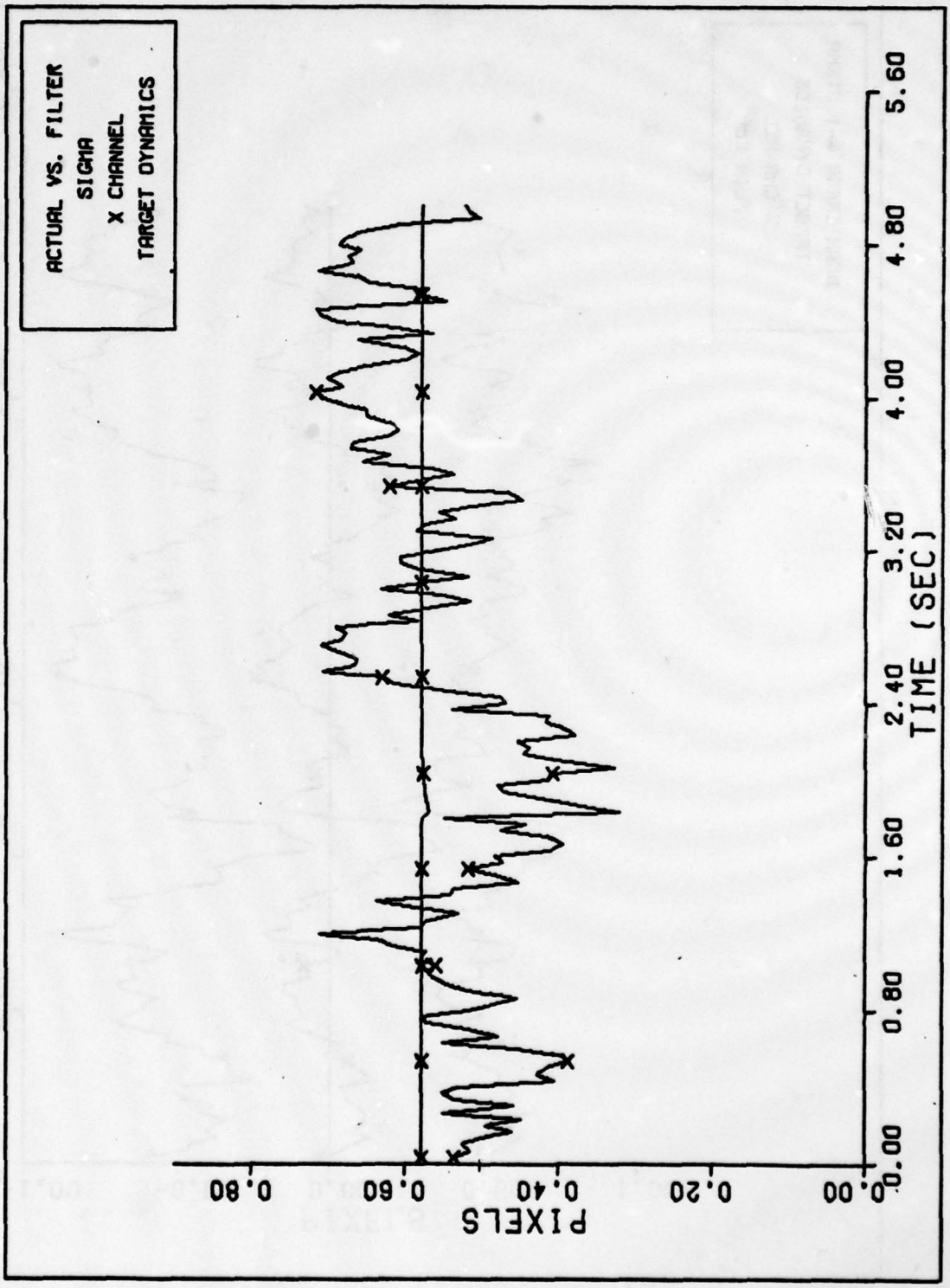
X CHANNEL ATMOSPHERICS ERROR

Figure 2c. Case 2 Performance Plot



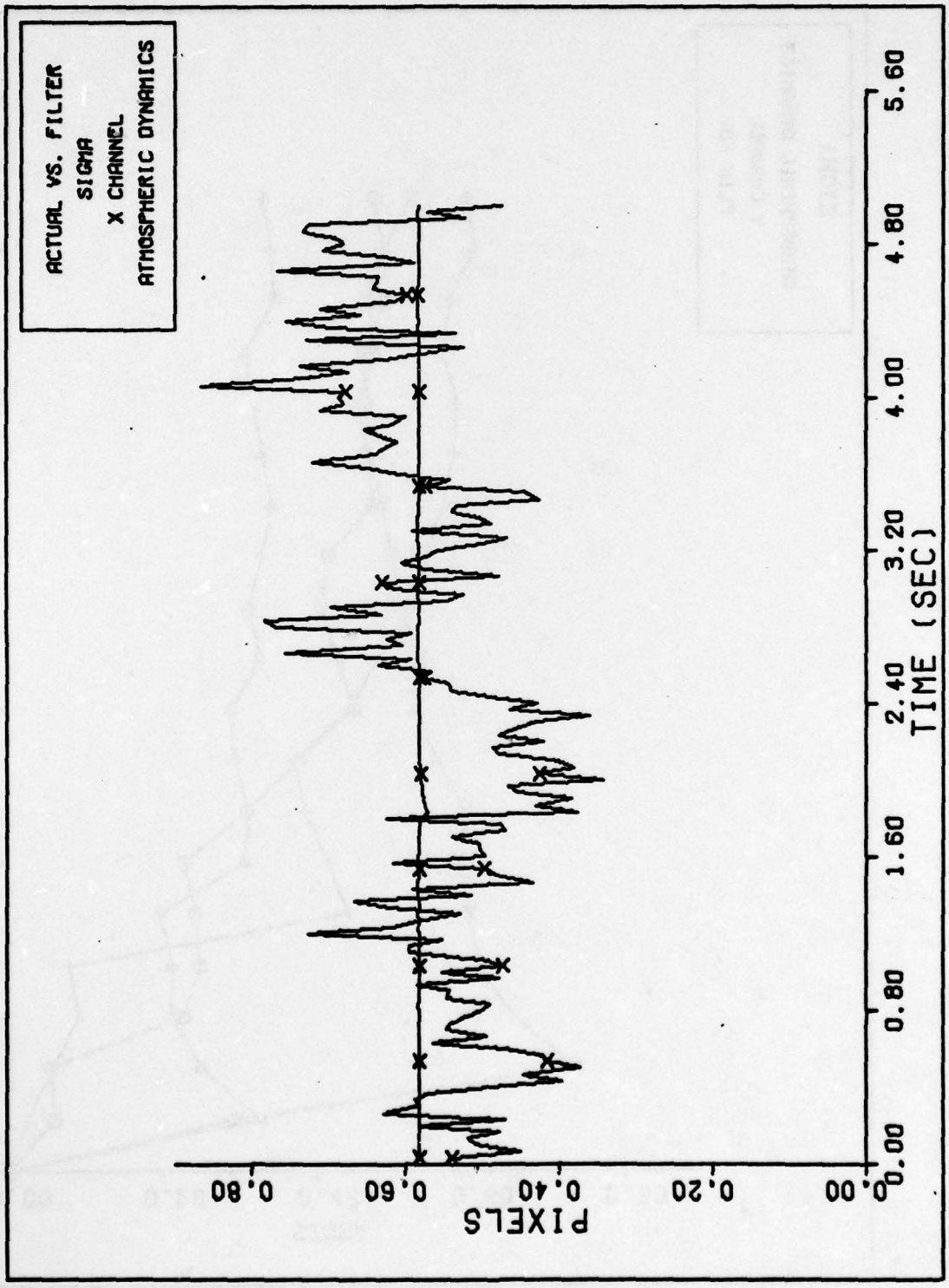
X CHANNEL DYNAMICS ERROR

Figure 2d. Case 2 Performance Plot



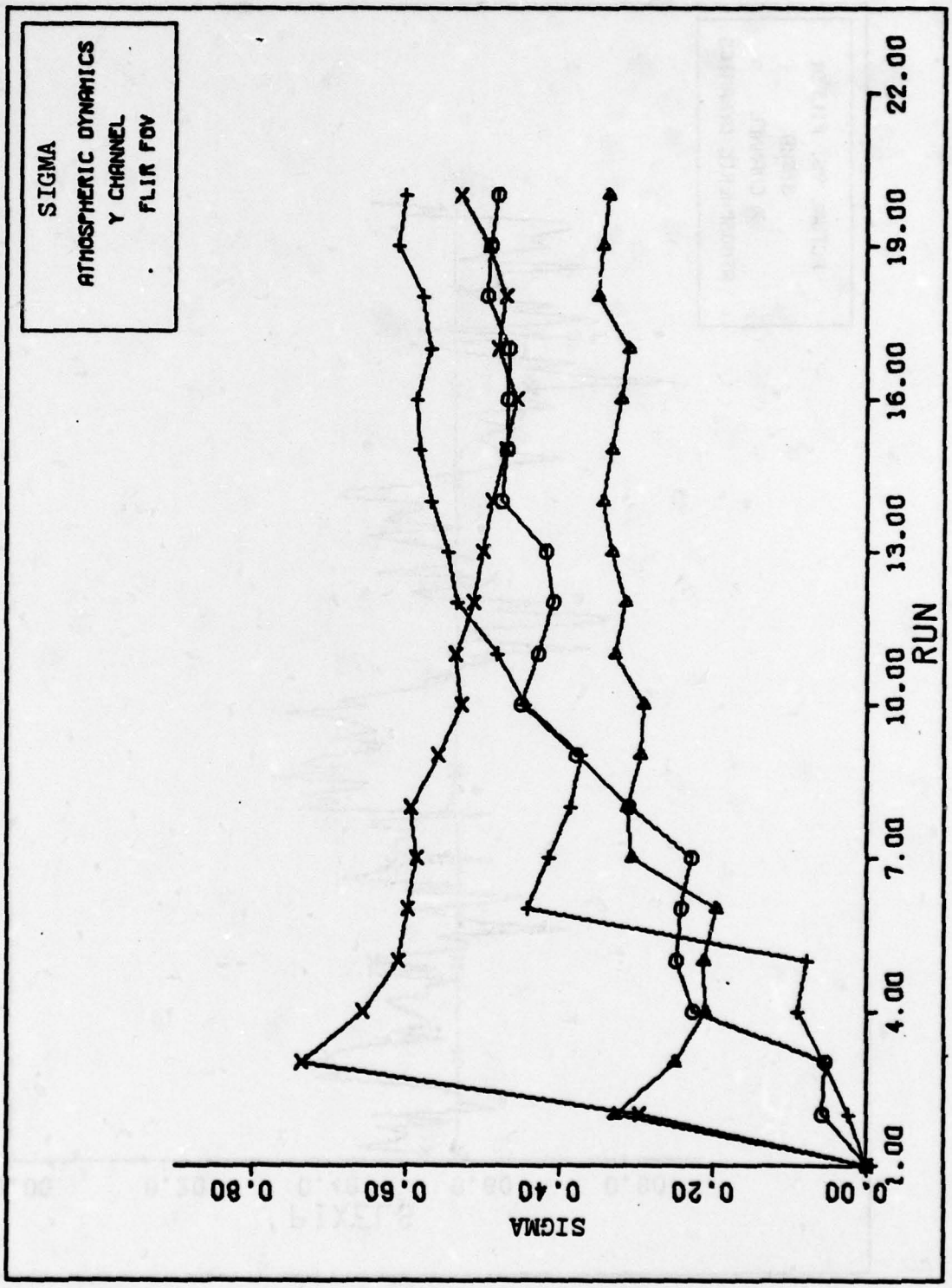
FILTER VS. ACTUAL SIGMA PLOT (S/N = 10)

Figure 2e. Case 2 Performance Plot



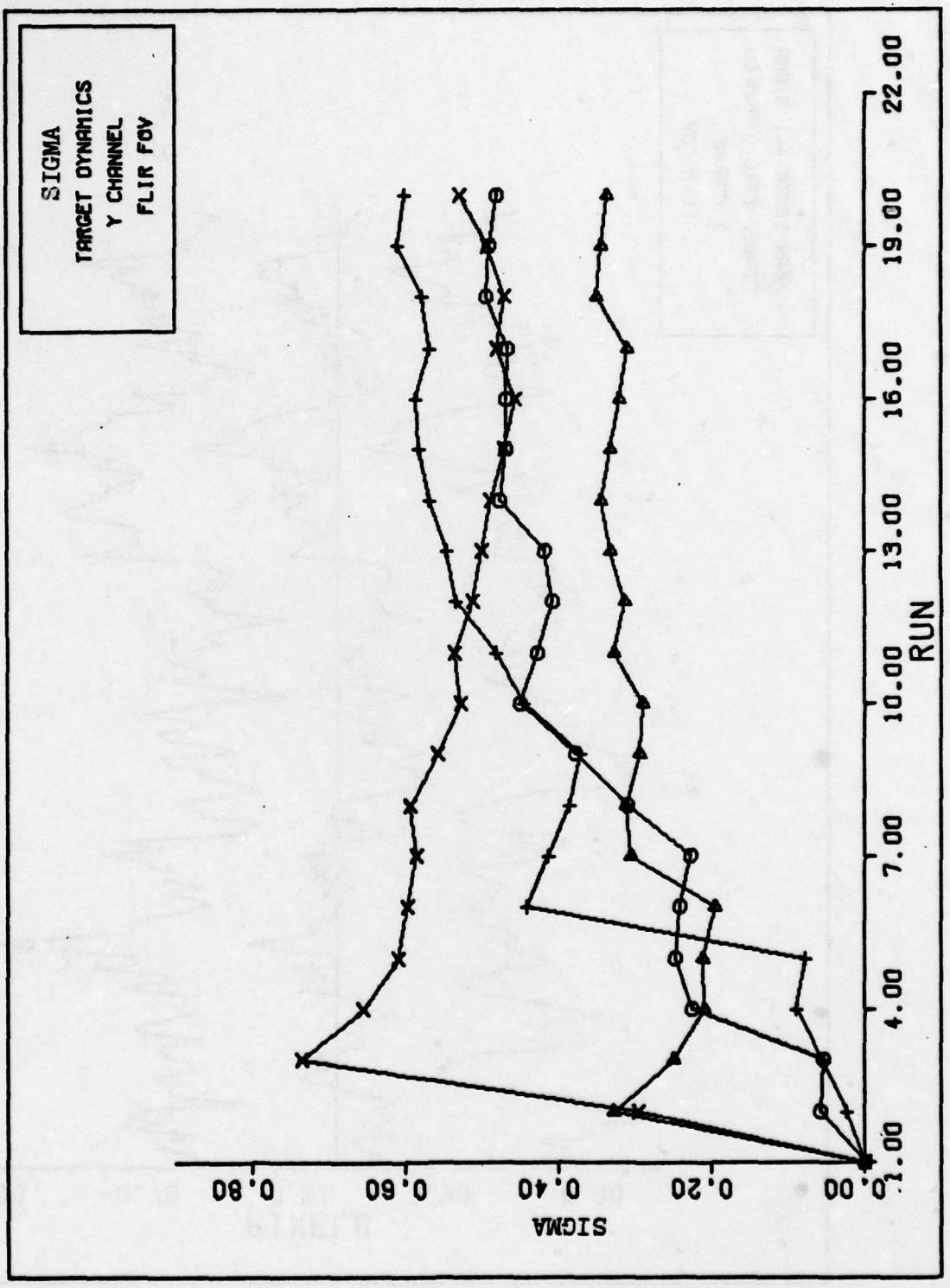
FILTER VS. ACTUAL SIGMA PLOT (S/N = 10)

Figure 2f. Case 2 Performance Plot



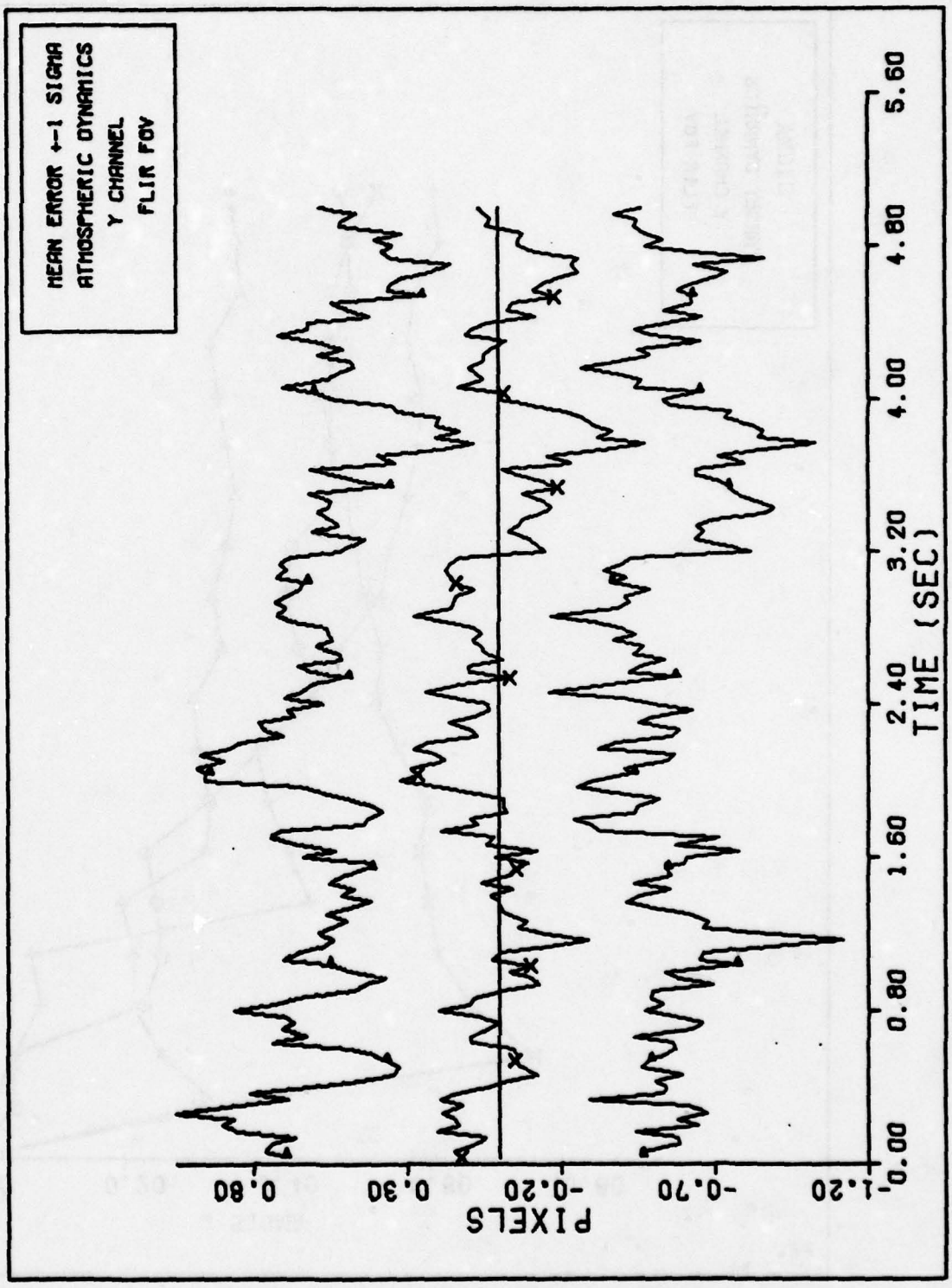
SIGMA CONVERGENCE

Figure 2g. Case 2 Performance Plot



SIGMA CONVERGENCE

Figure 2h. Case 2 Performance Plot



Y CHANNEL ATMOSPHERICS ERROR (S/N = 10)

Figure 2i. Case 2 Performance Plot

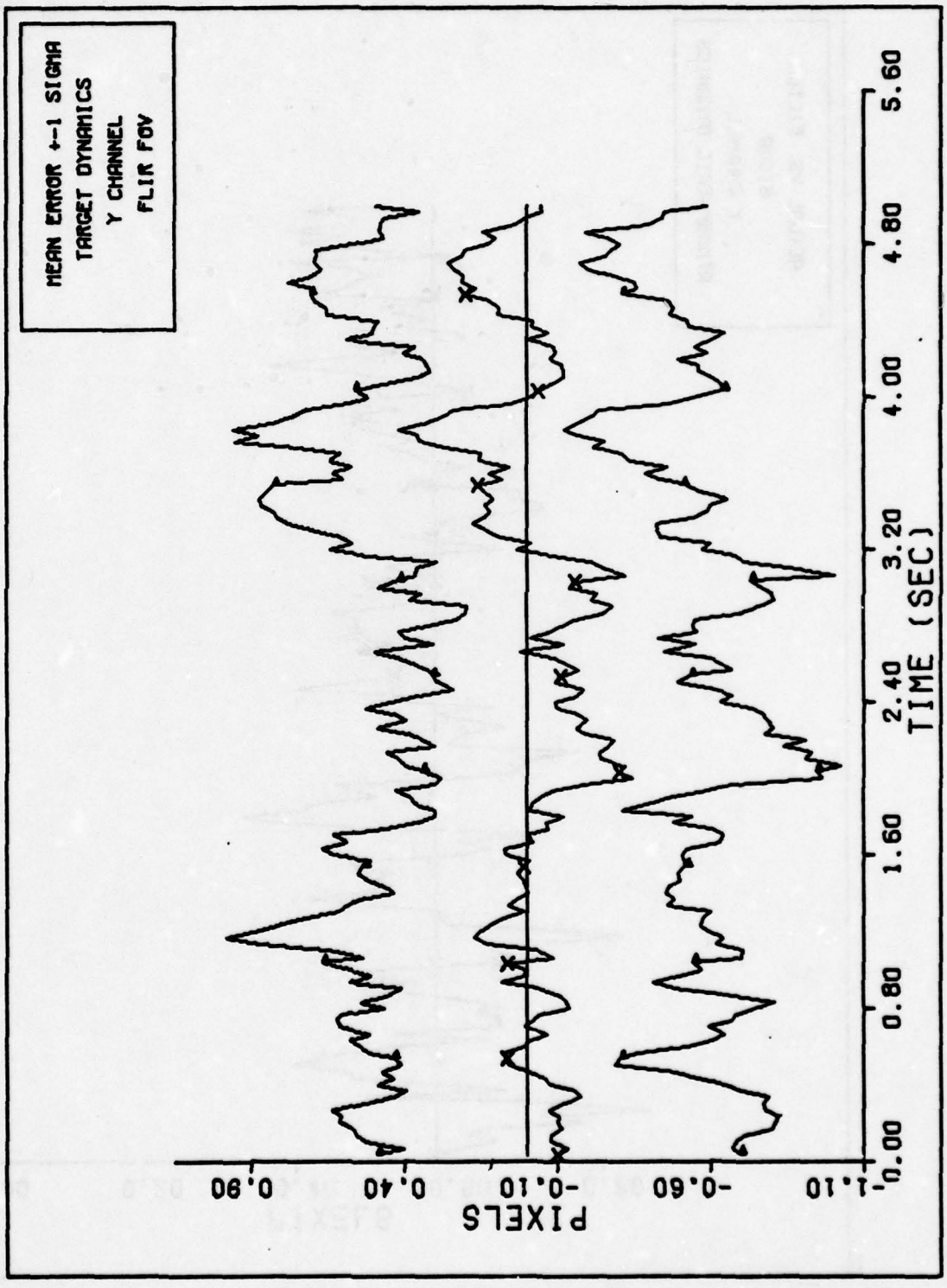
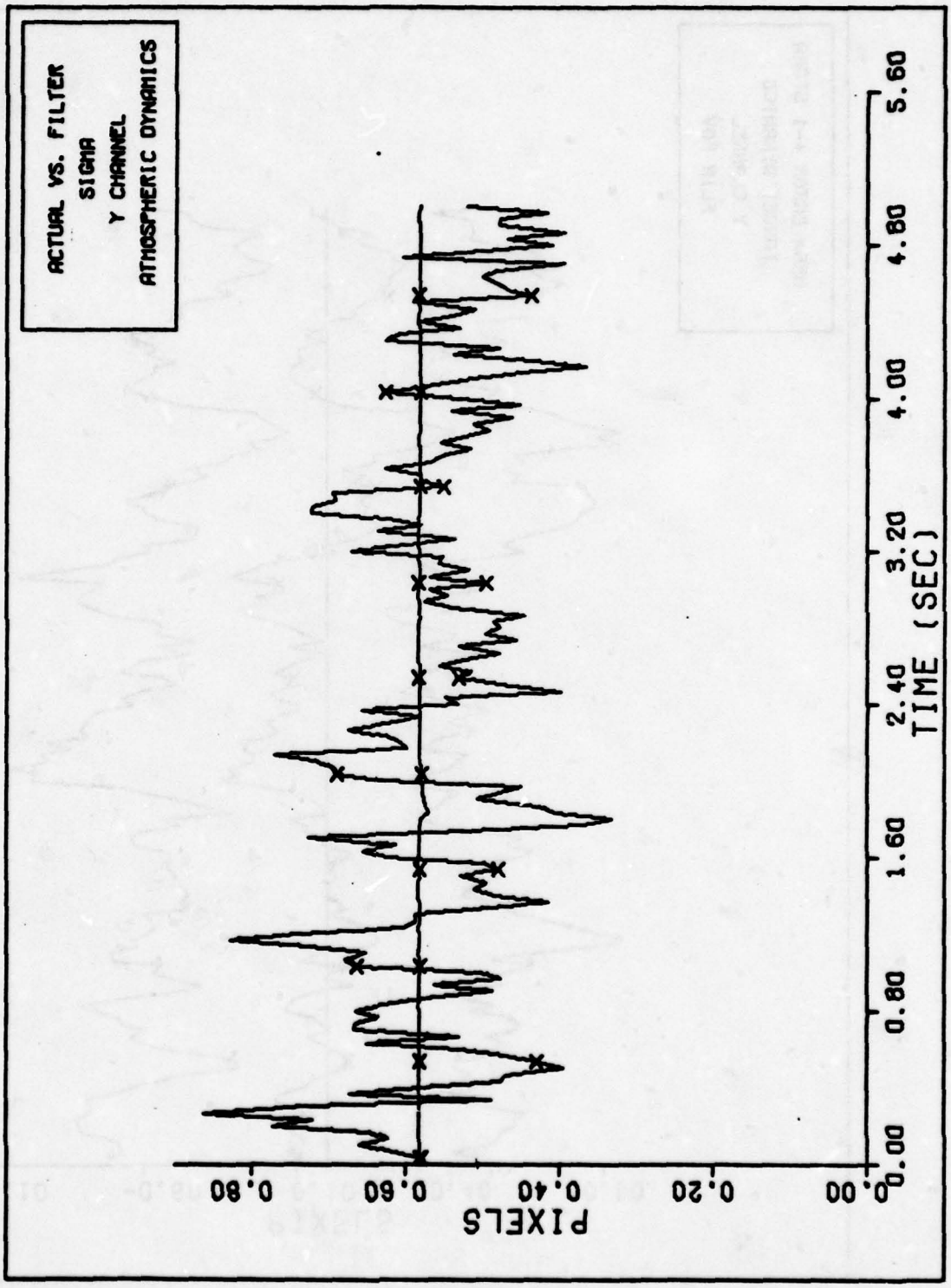


Figure 2j. Case 2 Performance Plot



FILTER VS. ACTUAL SIGMA PLOT (S/N = 10)

Figure 2k. Case 2 Performance Plot

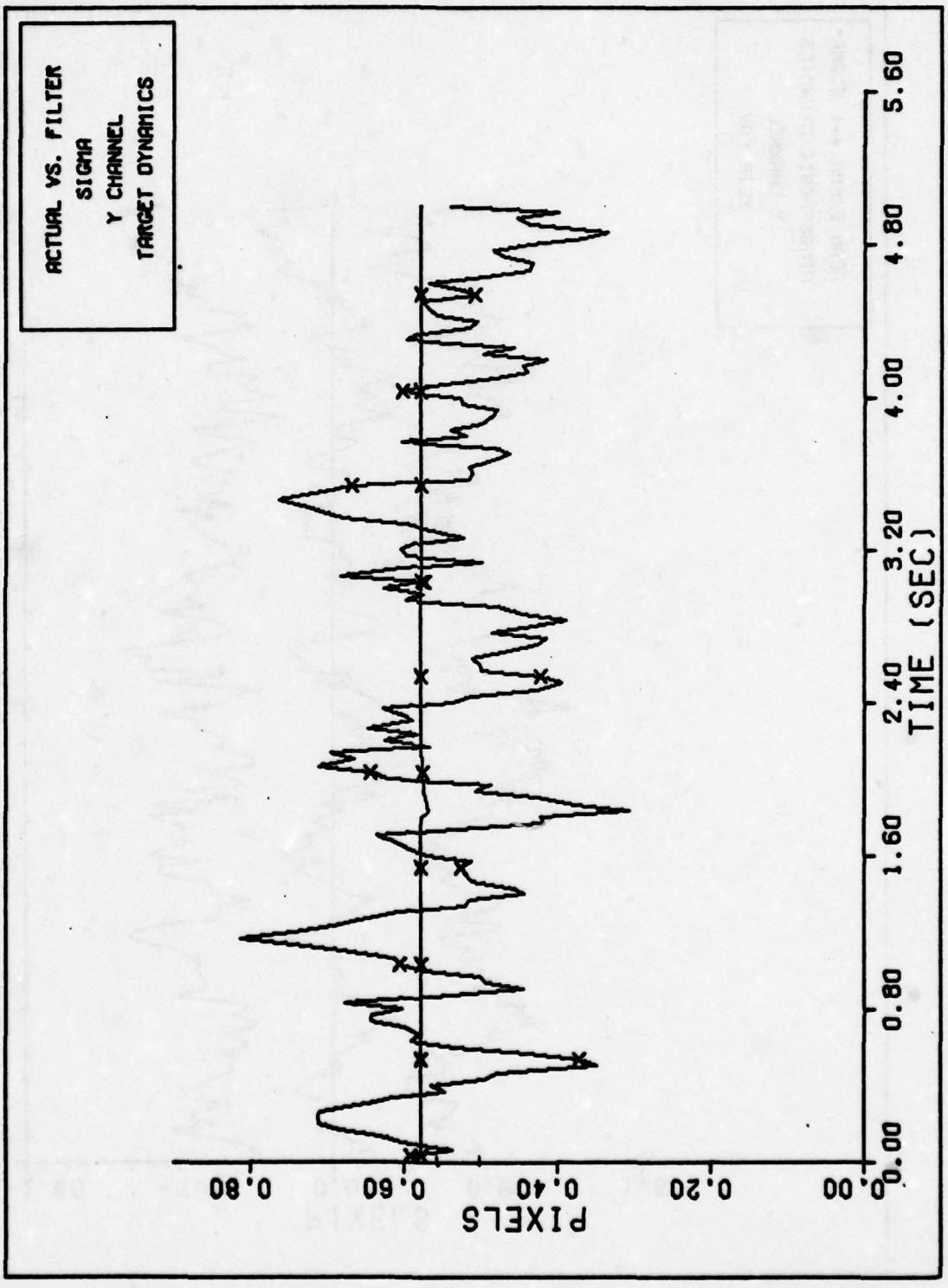
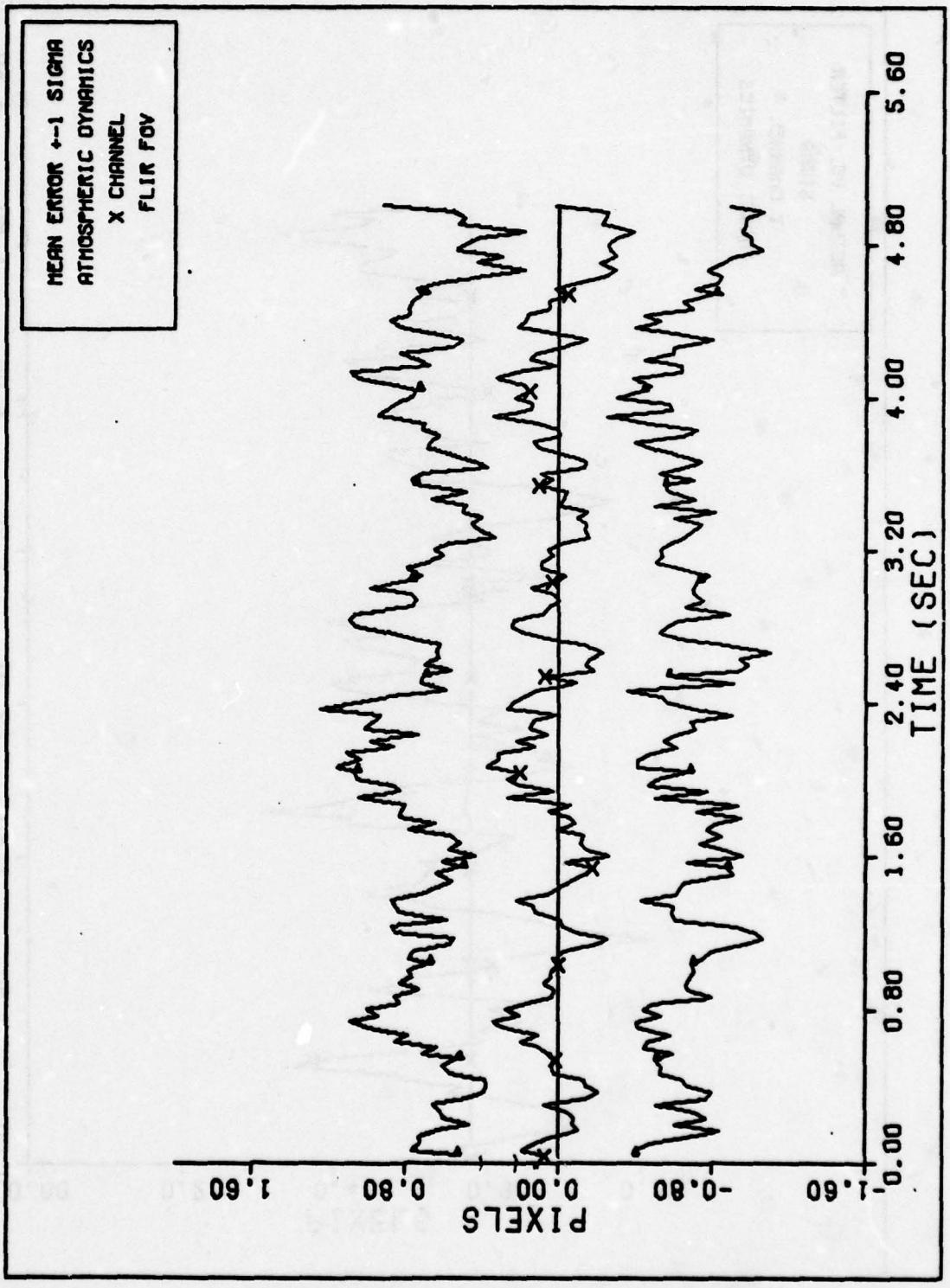
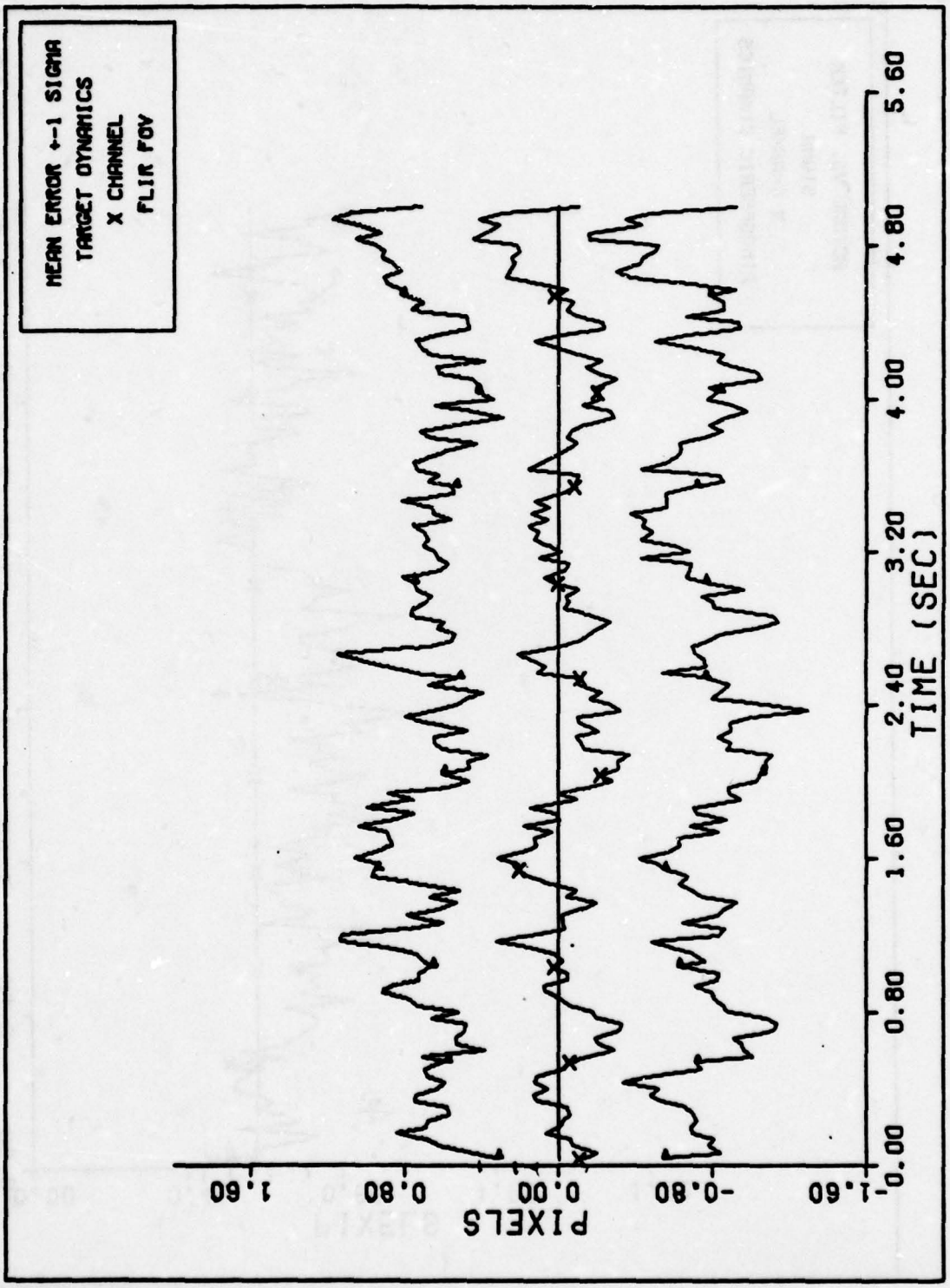


Figure 2ℓ. Case 2 Performance Plot



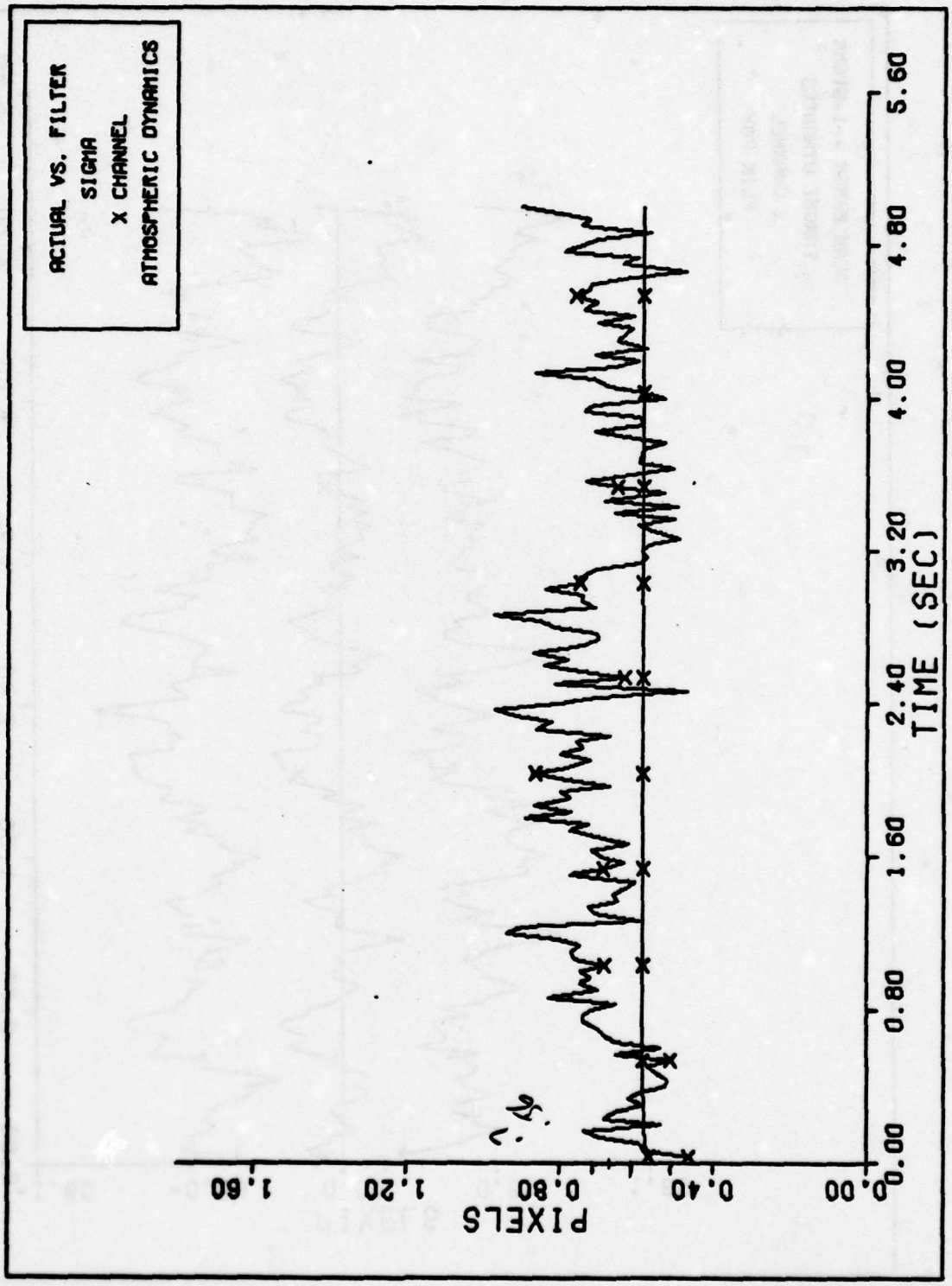
X CHANNEL ATMOSPHERICS ERROR (S/N= 10)

Figure 3a. Case 3 Performance Plot



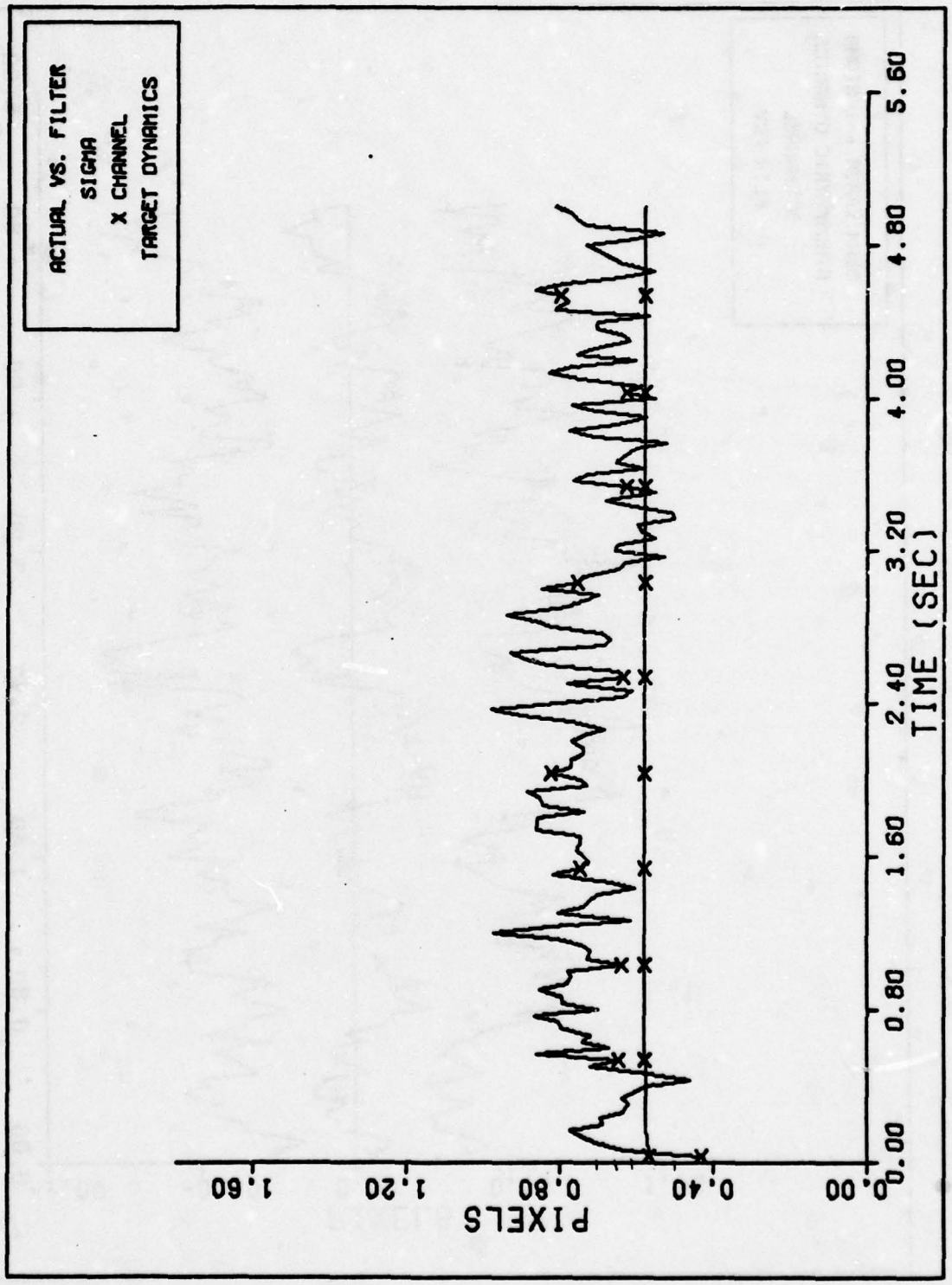
X CHANNEL DYNAMICS ERROR (S/N= 10)

Figure 3b. Case 3 Performance Plot



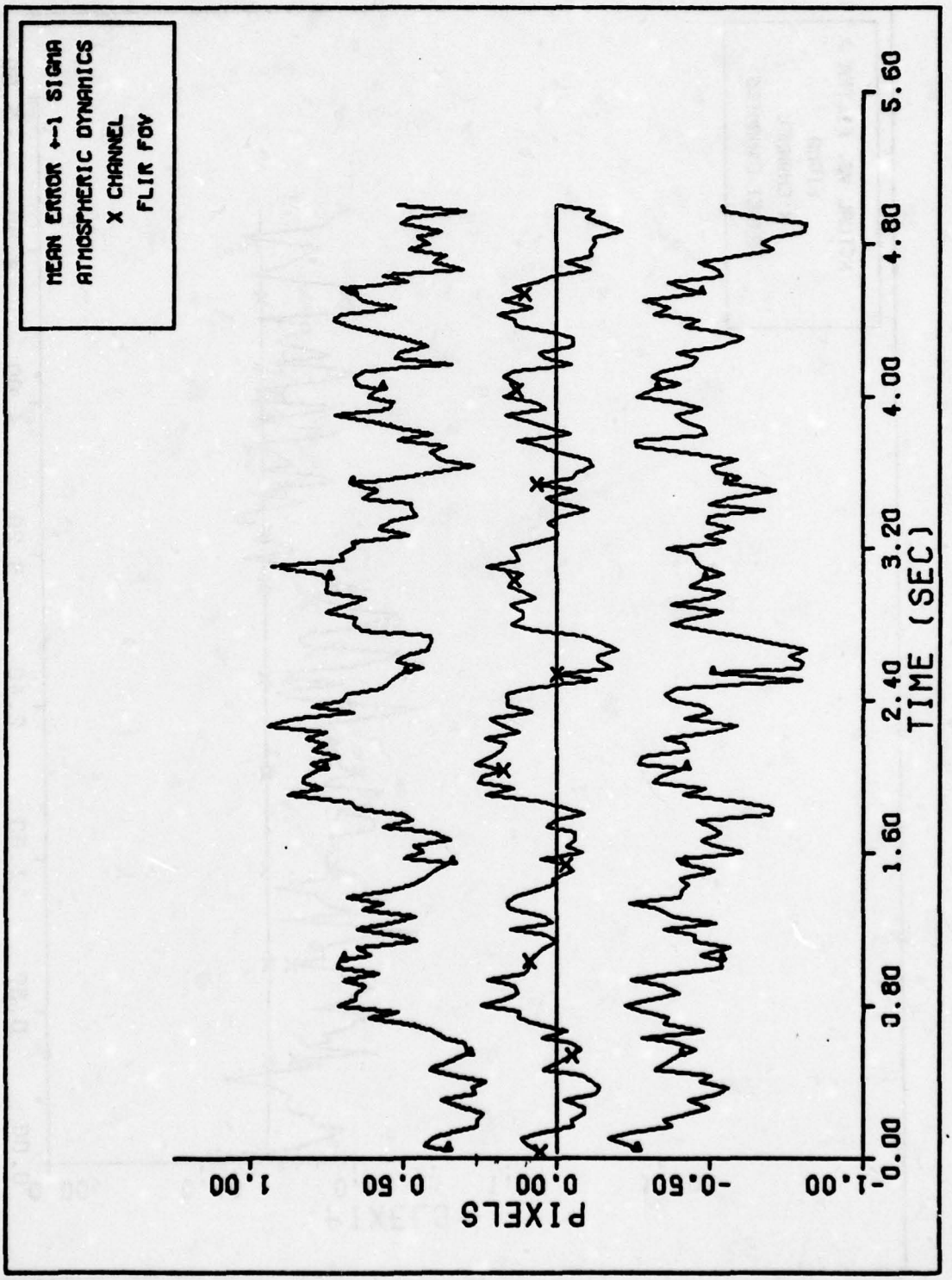
FILTER VS. ACTUAL SIGMA PLOT (S/N = 10)

Figure 3c. Case 3 Performance Plot



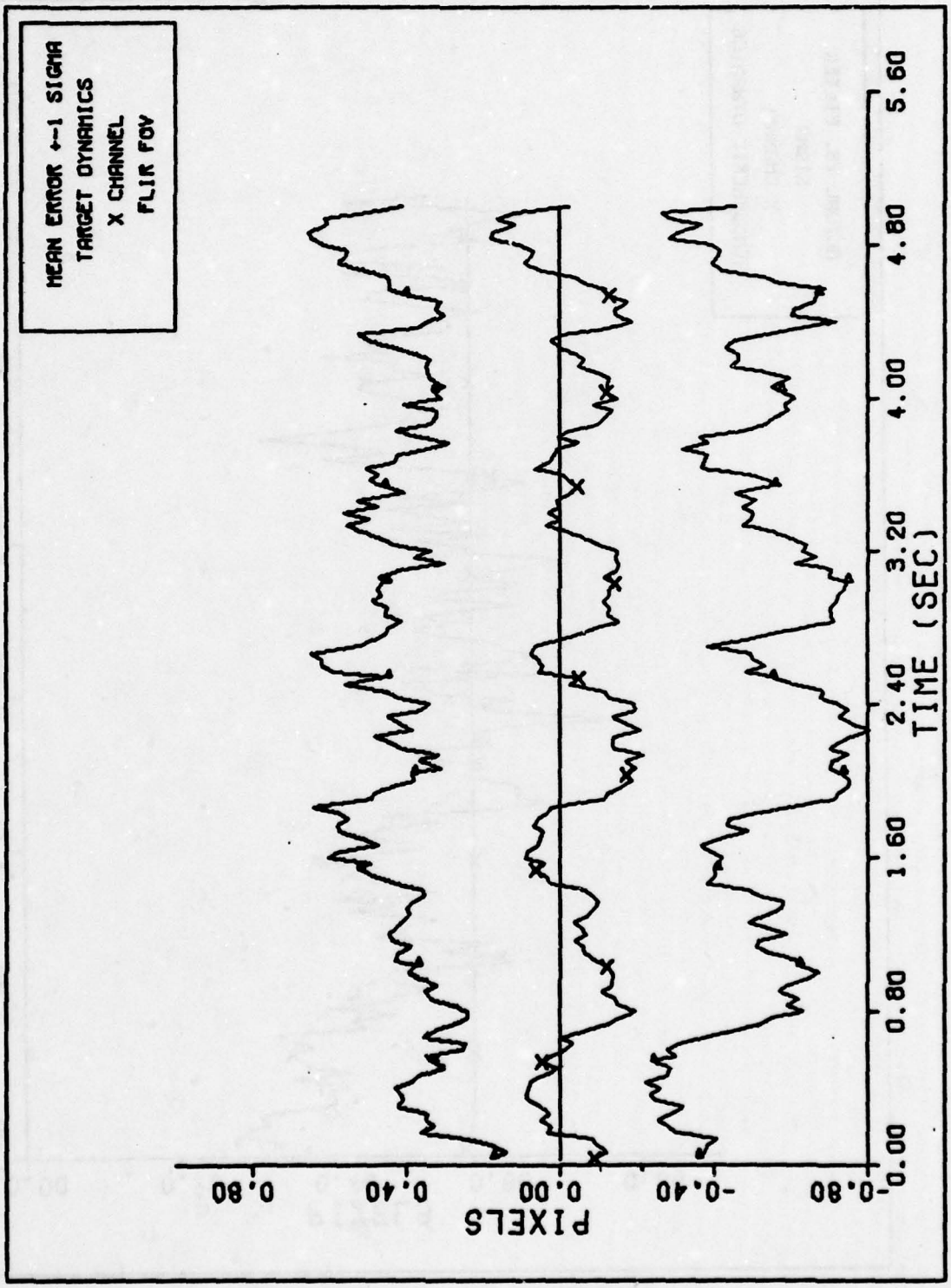
FILTER VS. ACTUAL SIGMA PLOT (S/N = 10)

Figure 3d. Case 3 Performance Plot



X CHANNEL ATMOSPHERICS ERROR (S/N= 10)

Figure 4a. Case 4 Performance Plot



X CHANNEL DYNAMICS ERROR (S/N= 10)

Figure 4b. Case 4 Performance Plot

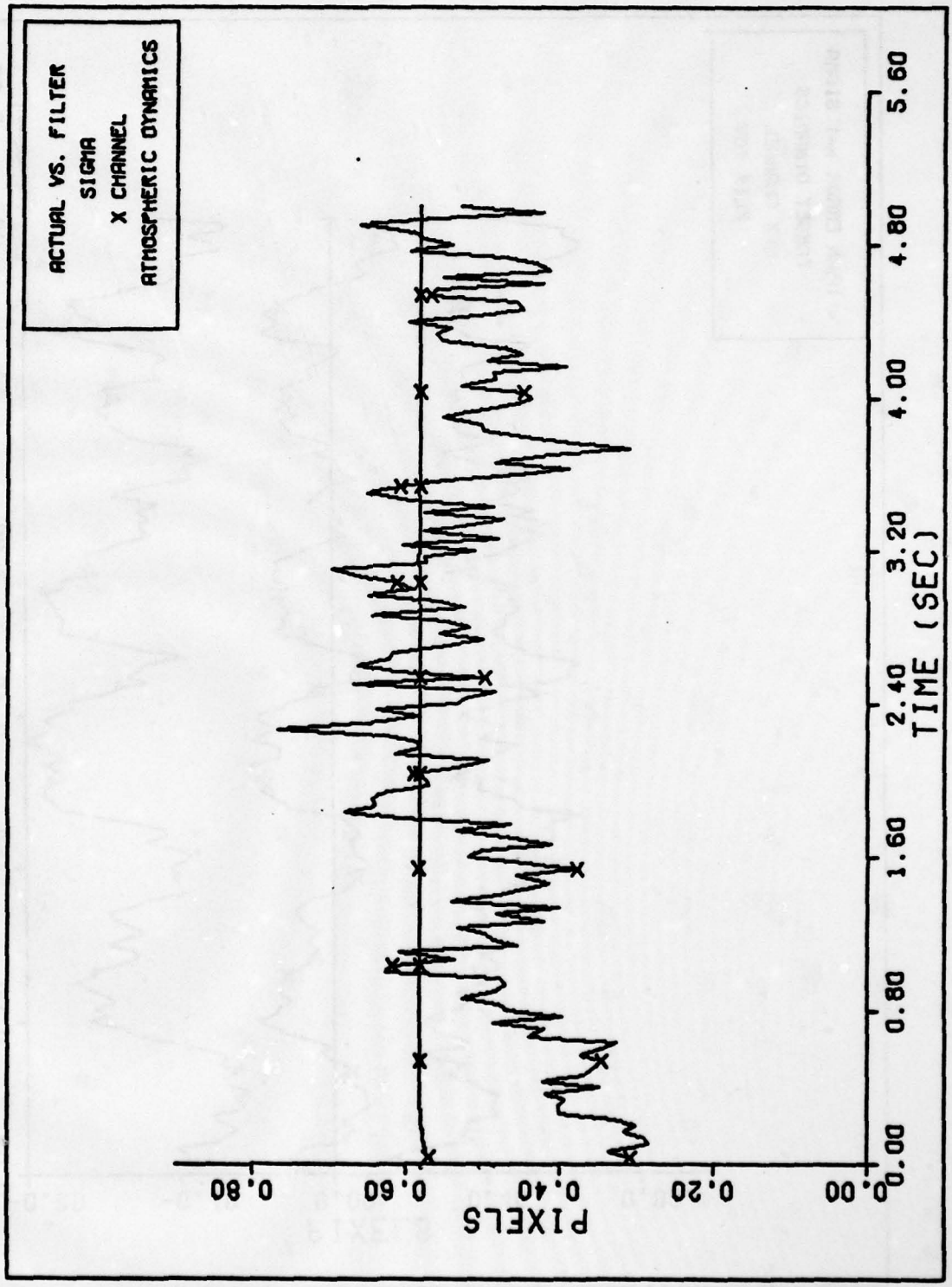
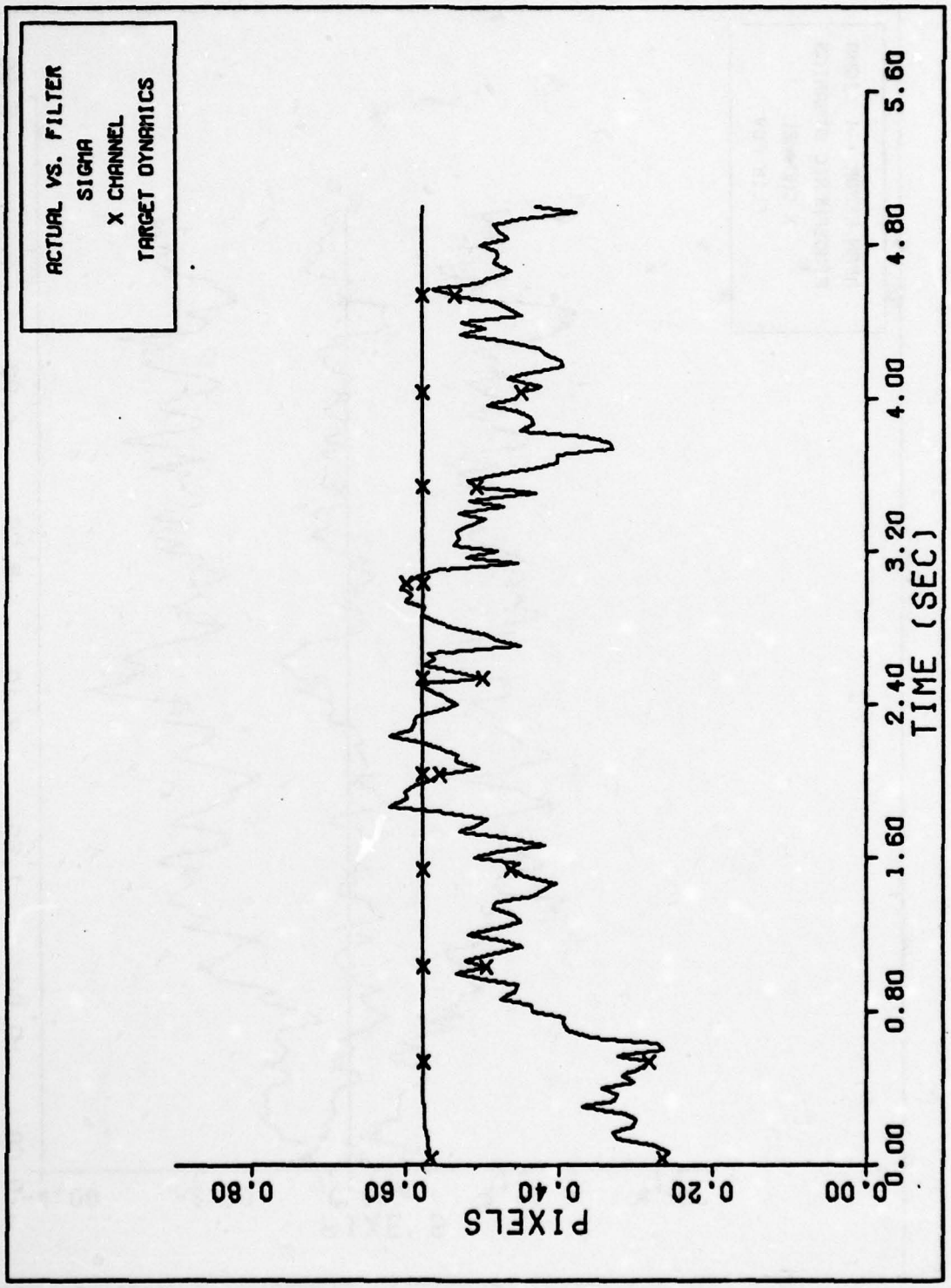


Figure 4c. Case 4 Performance Plot



FILTER VS. ACTUAL SIGMA PLOT (S/N = 10)

Figure 4d. Case 4 Performance Plot

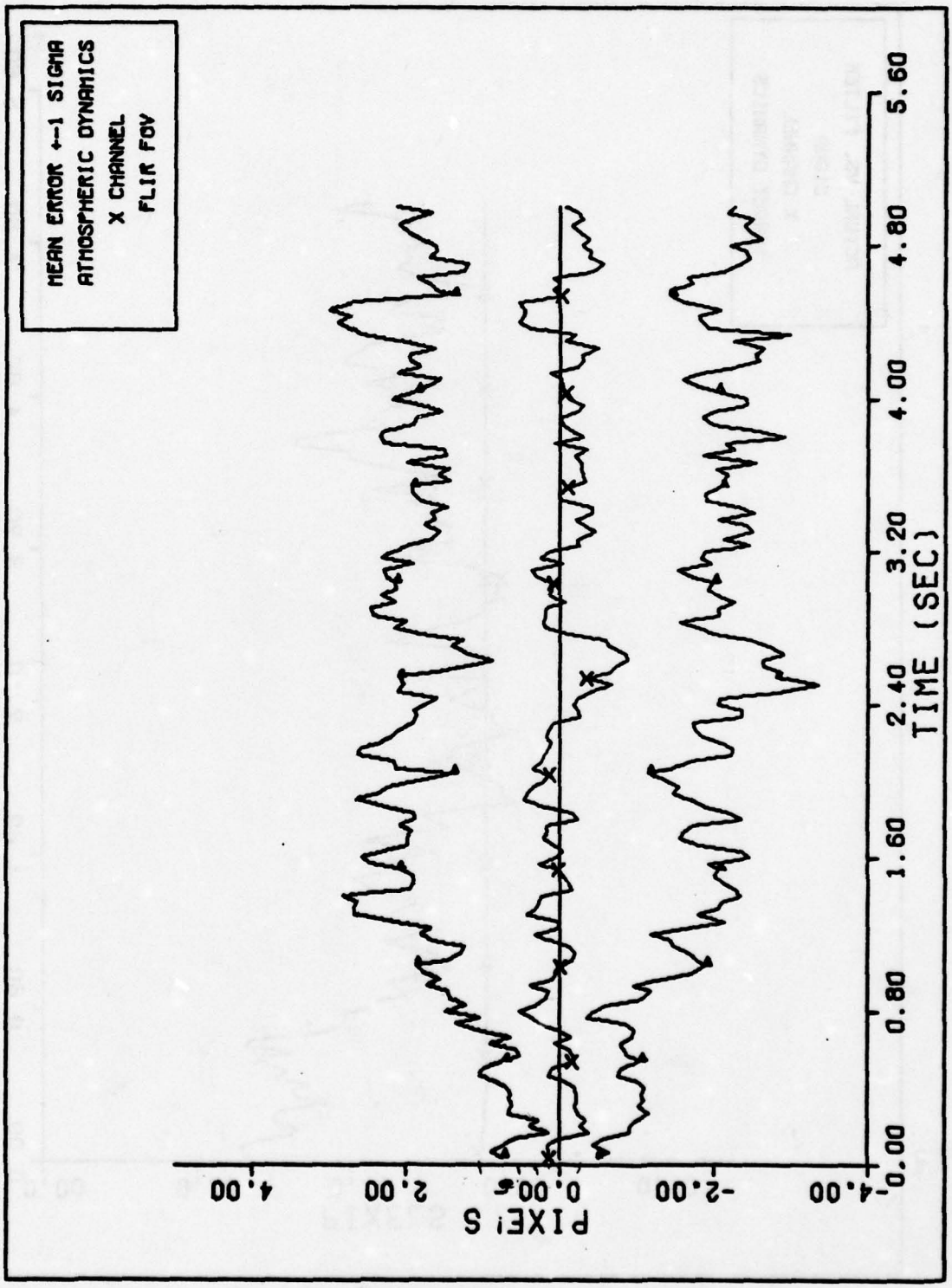
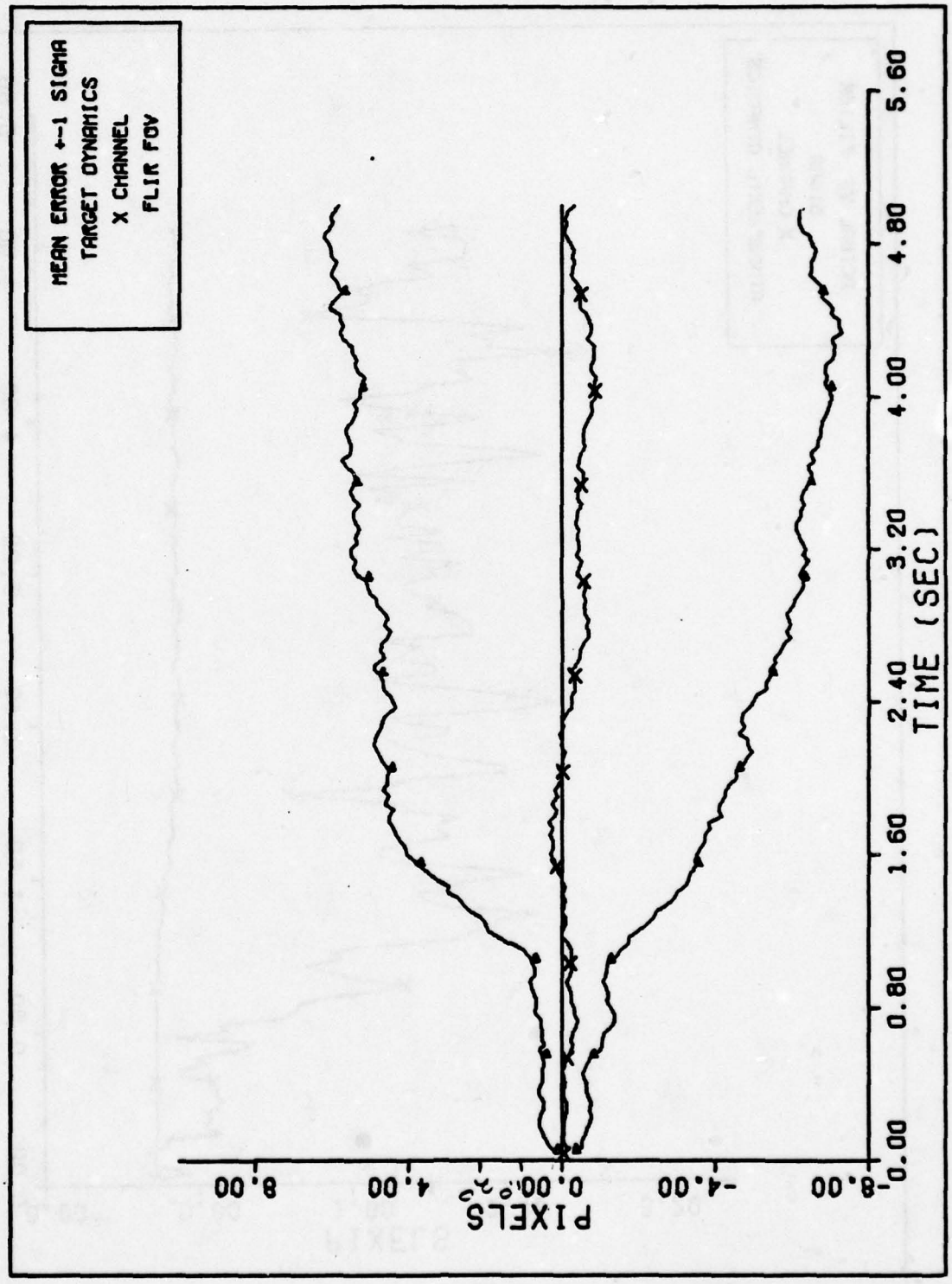
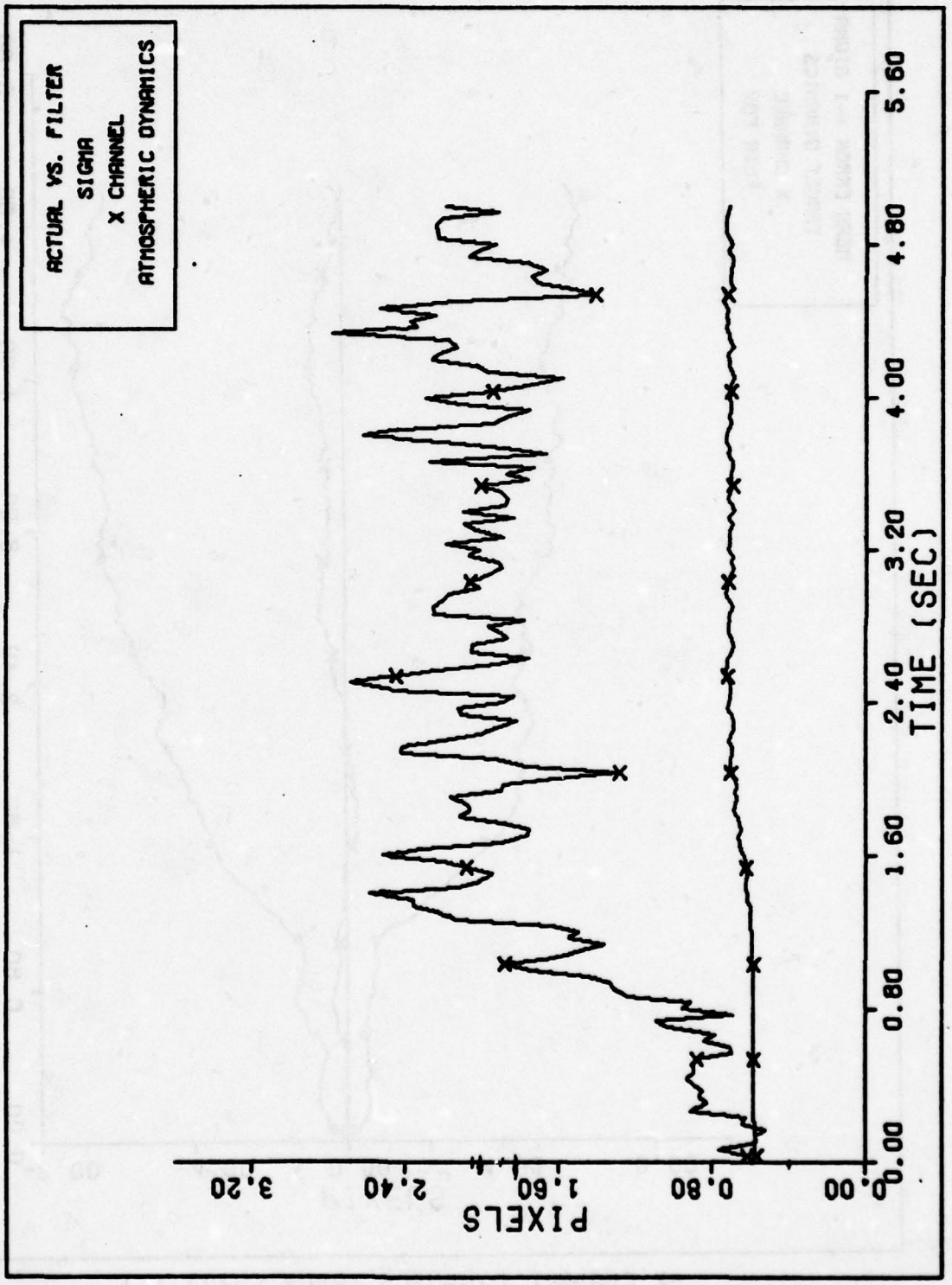


Figure 5a. Case 5 Performance Plot



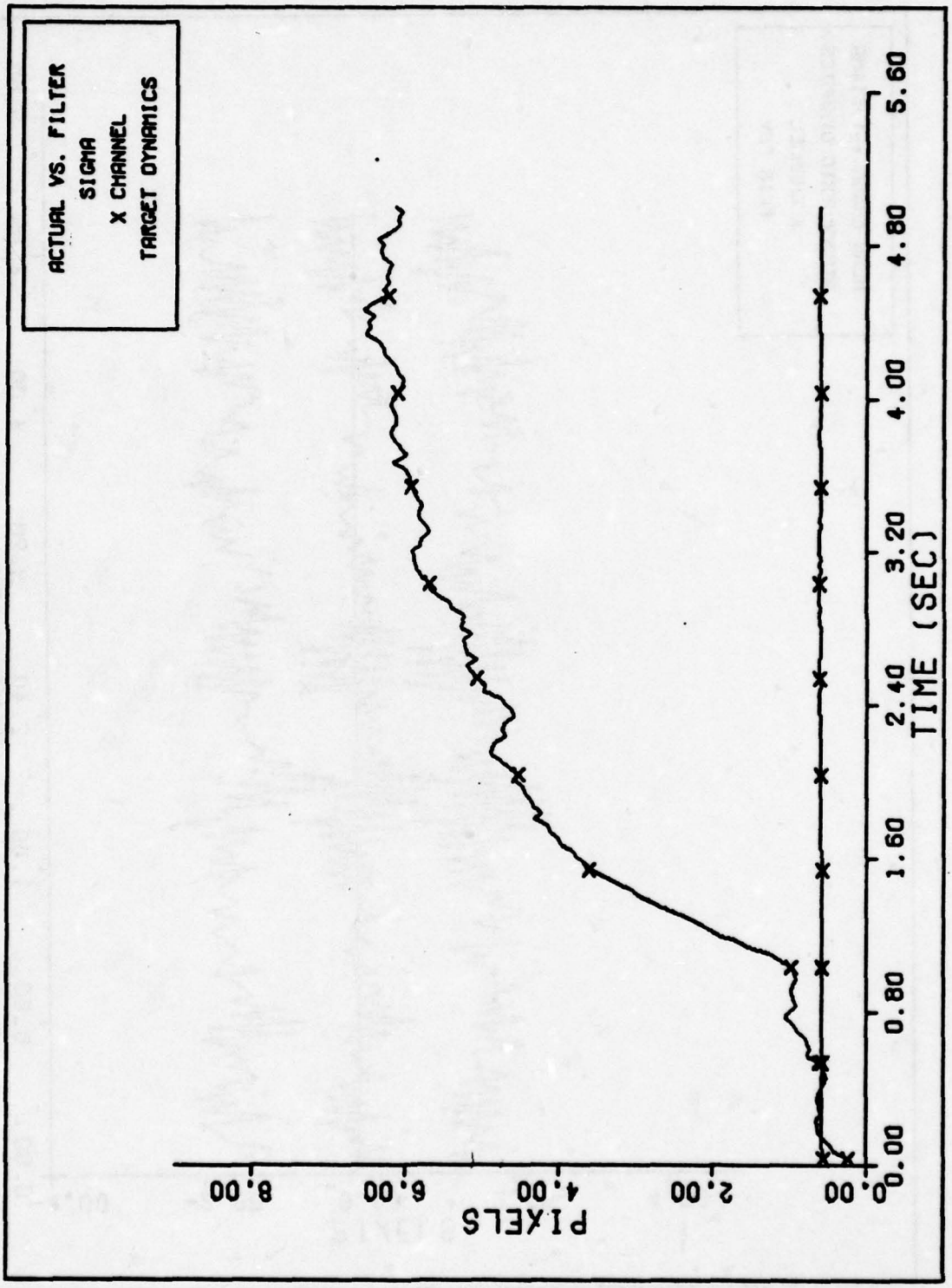
X CHANNEL DYNAMICS ERROR (S/N= 1)

Figure 5b. Case 5 Performance Plot



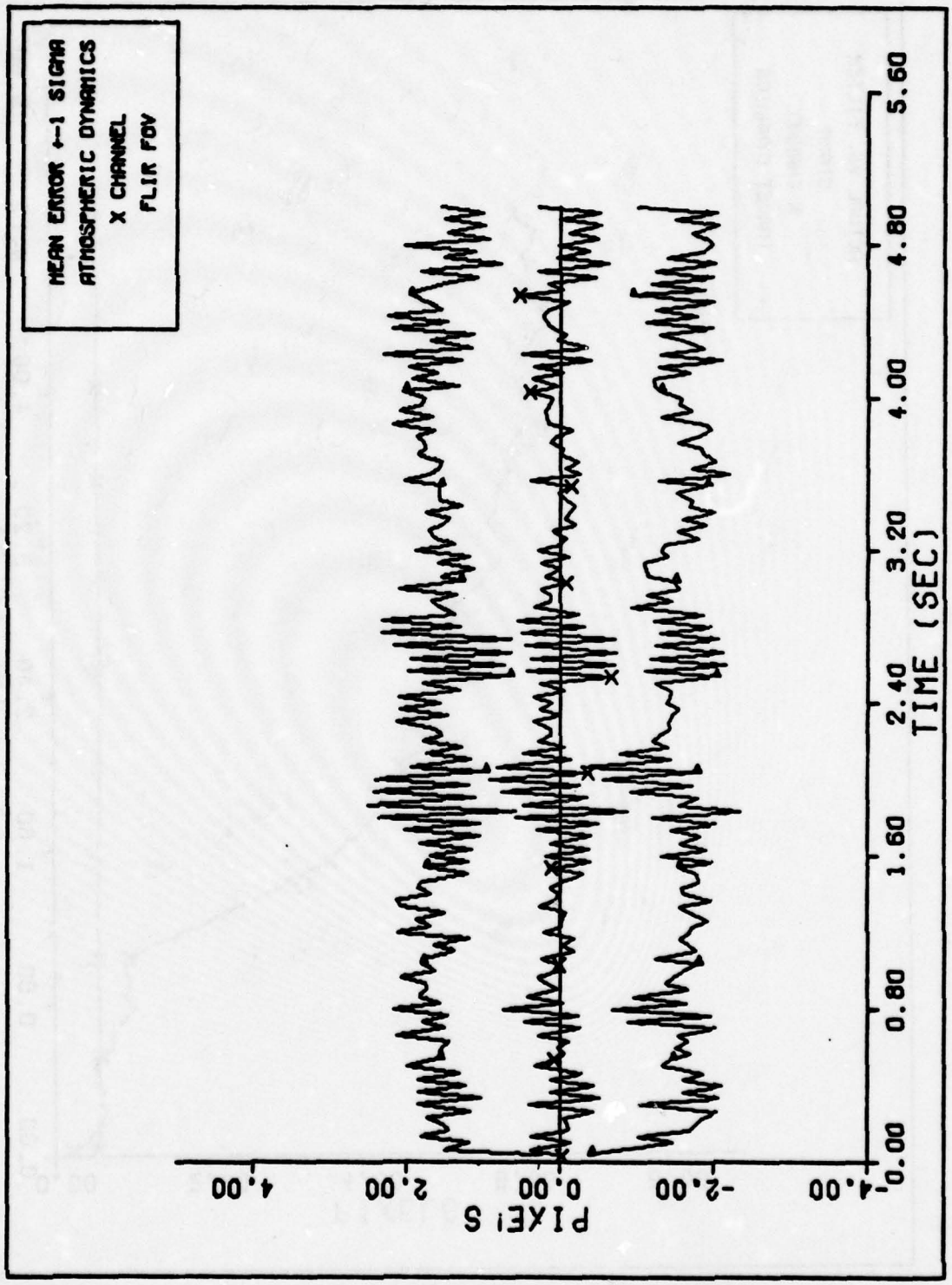
FILTER VS. ACTUAL SIGMA PLOT (S/N = 1)

Figure 5c. Case 5 Performance Plot



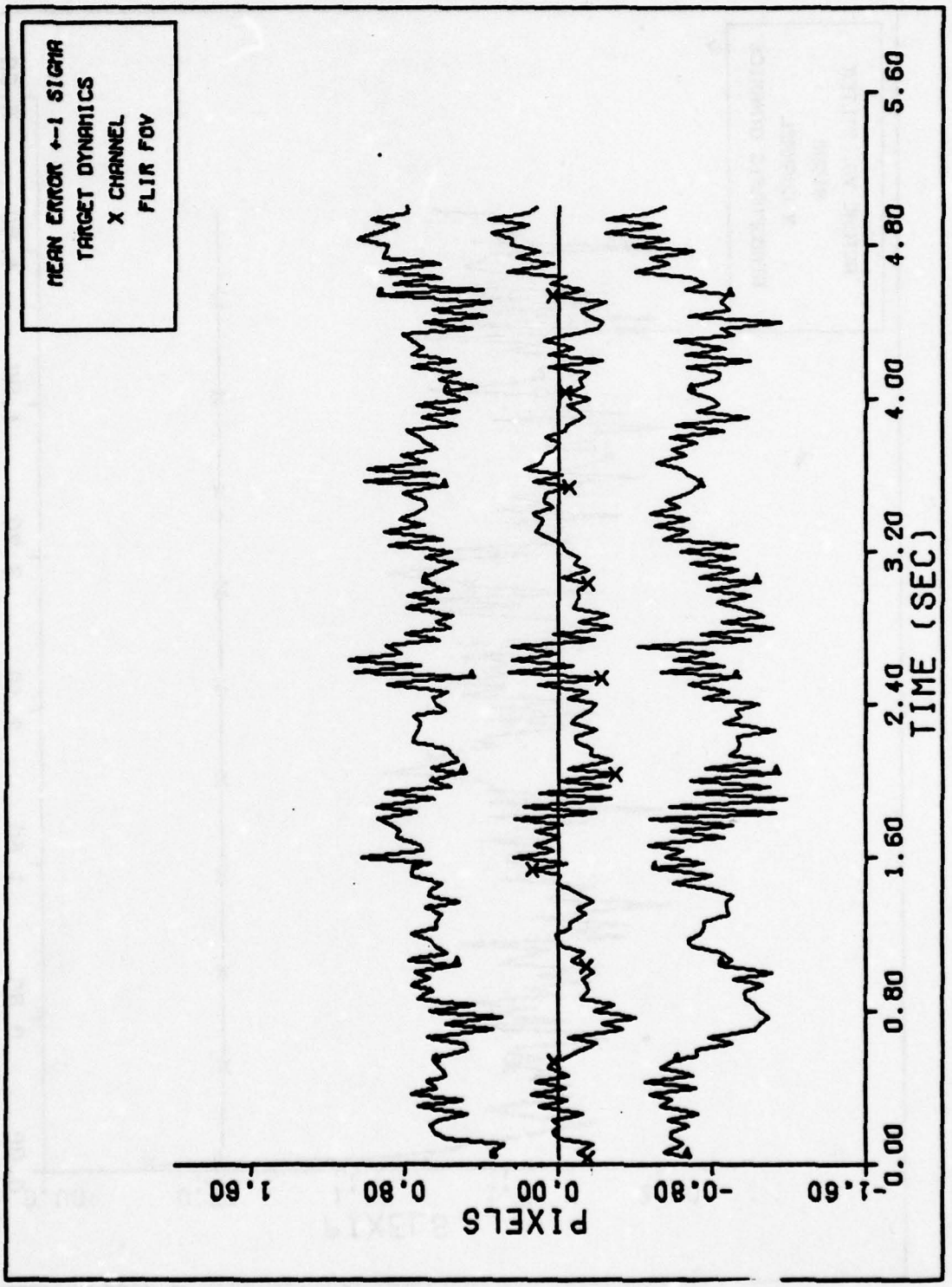
FILTER VS. ACTUAL SIGMA PLOT (S/N = 1)

Figure 5d. Case 5 Performance Plot



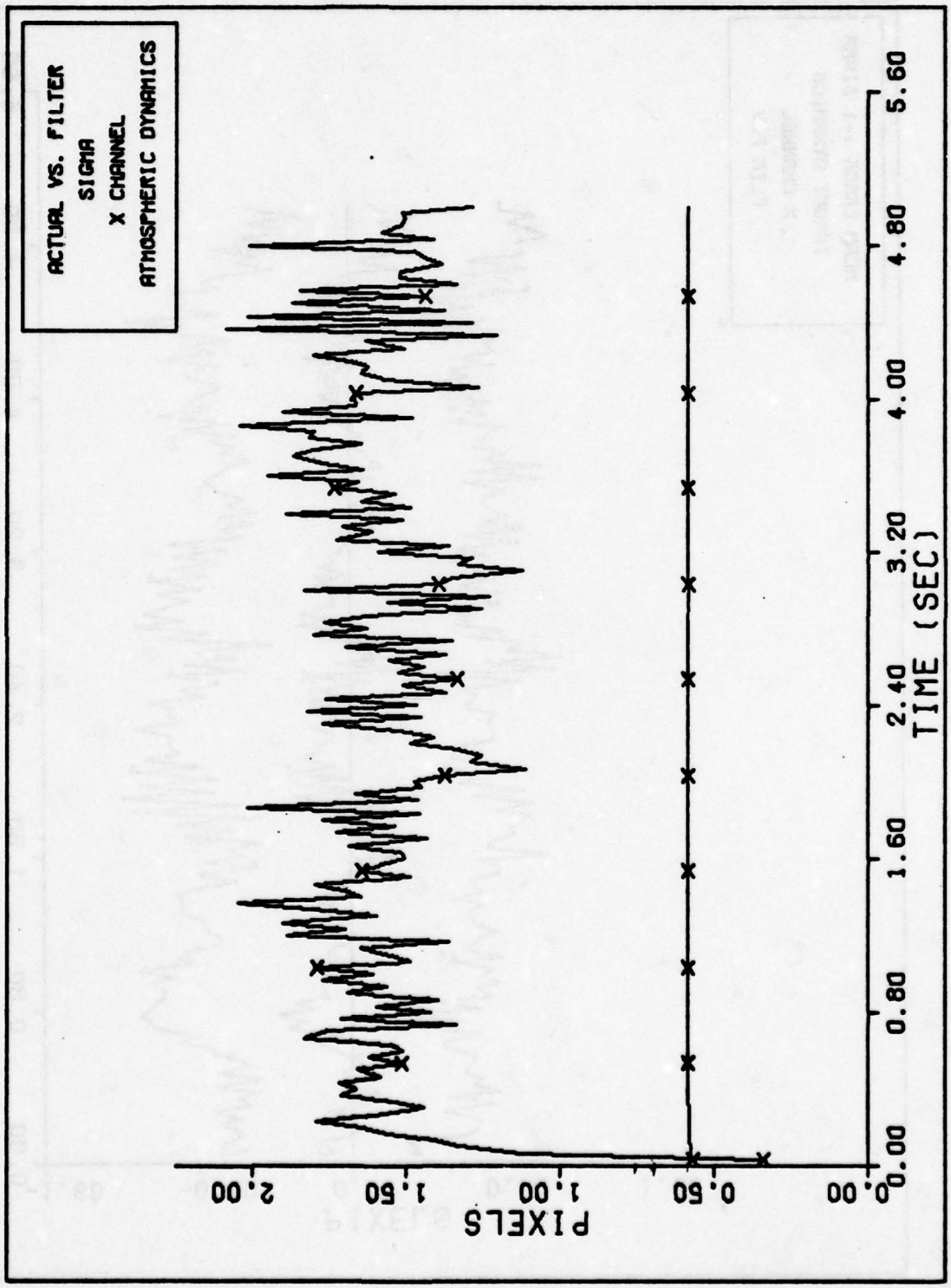
X CHANNEL ATMOSPHERICS ERROR

Figure 6a. Case 6 Performance Plot



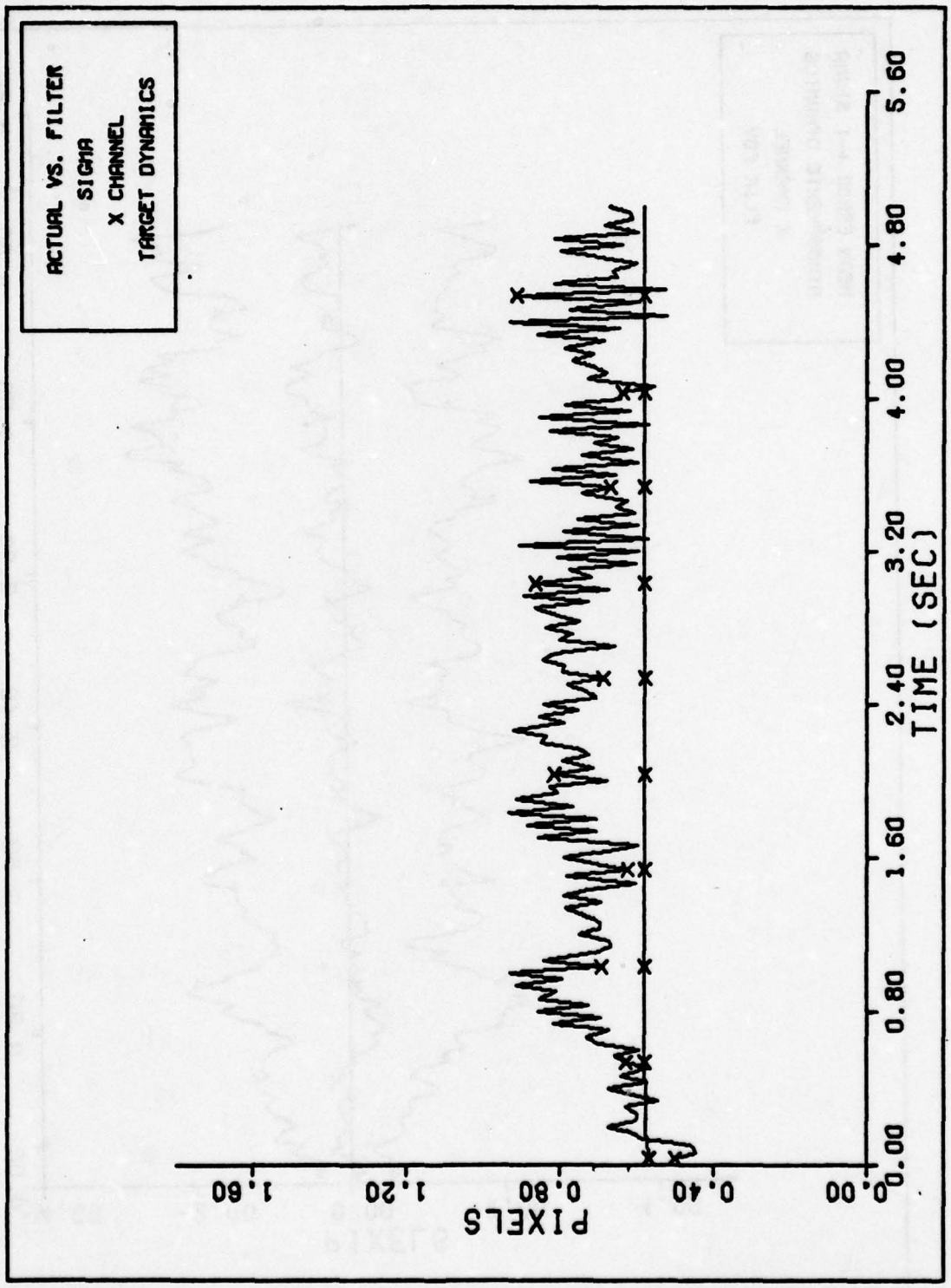
X CHANNEL DYNAMICS ERROR

Figure 6b. Case 6 Performance Plot



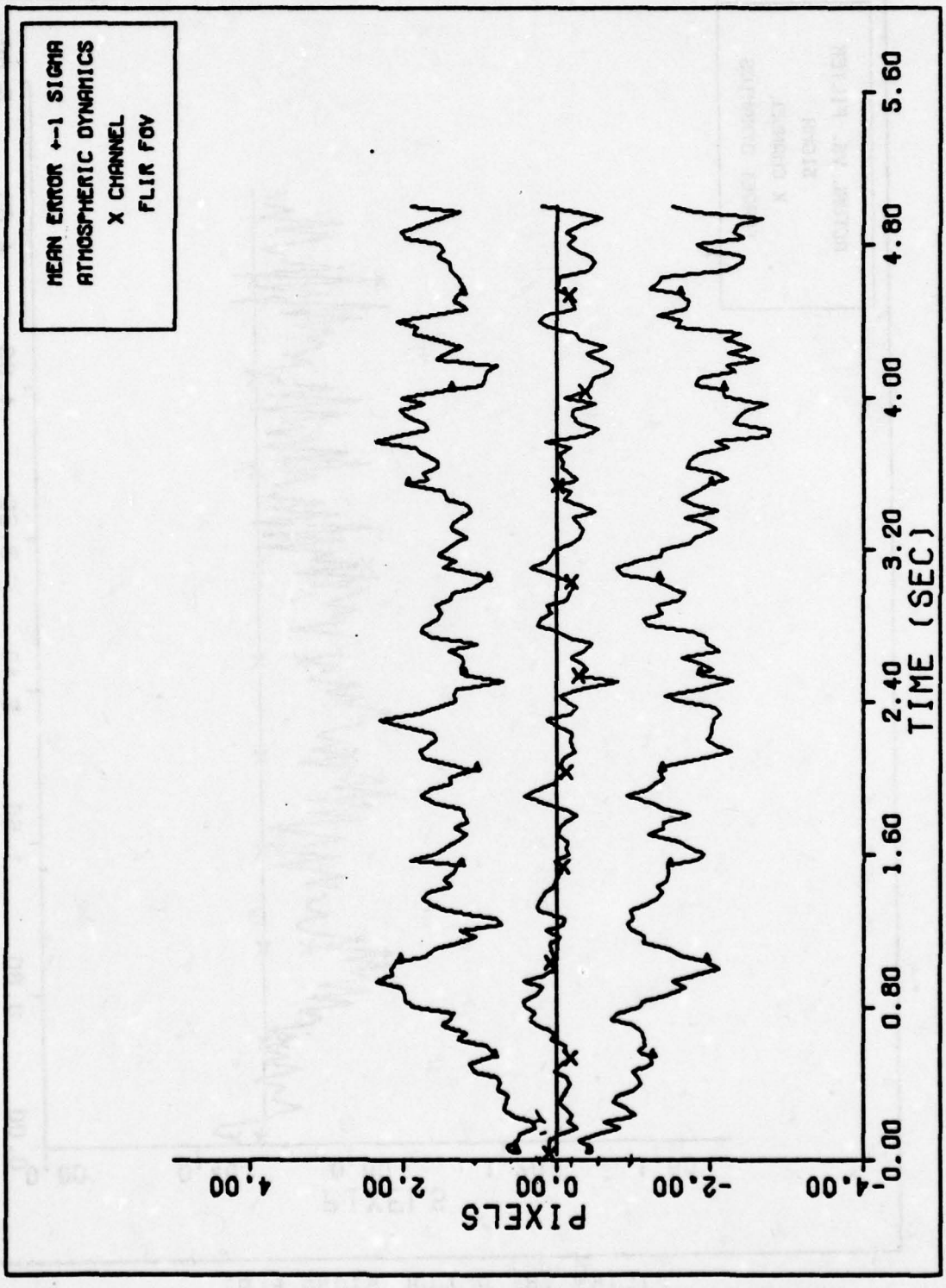
FILTER VS. ACTUAL SIGMA PLOT

Figure 6c. Case 6 Performance Plot



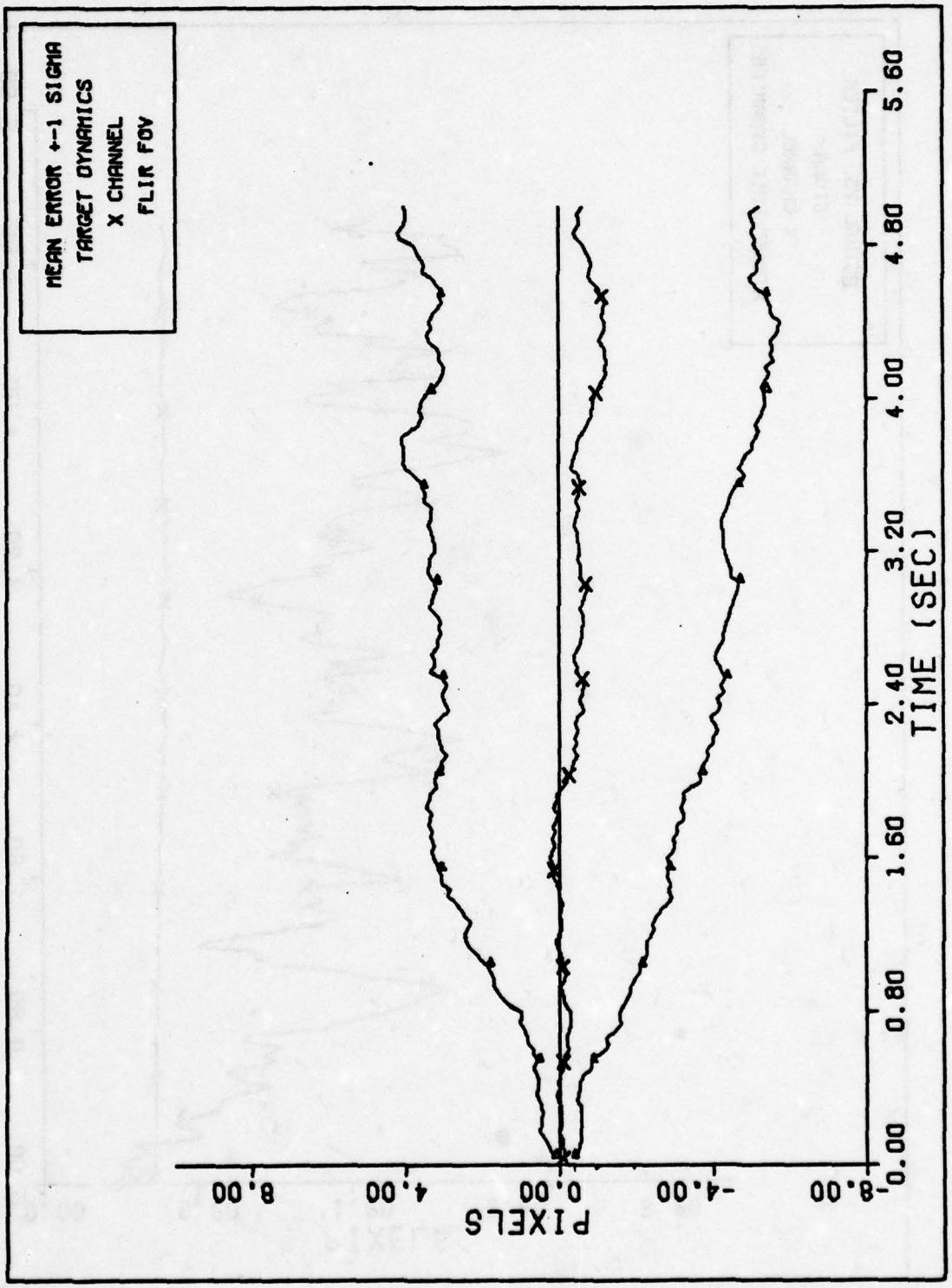
FILTER VS. ACTUAL SIGMA PLOT

Figure 6d. Case 6 Performance Plot



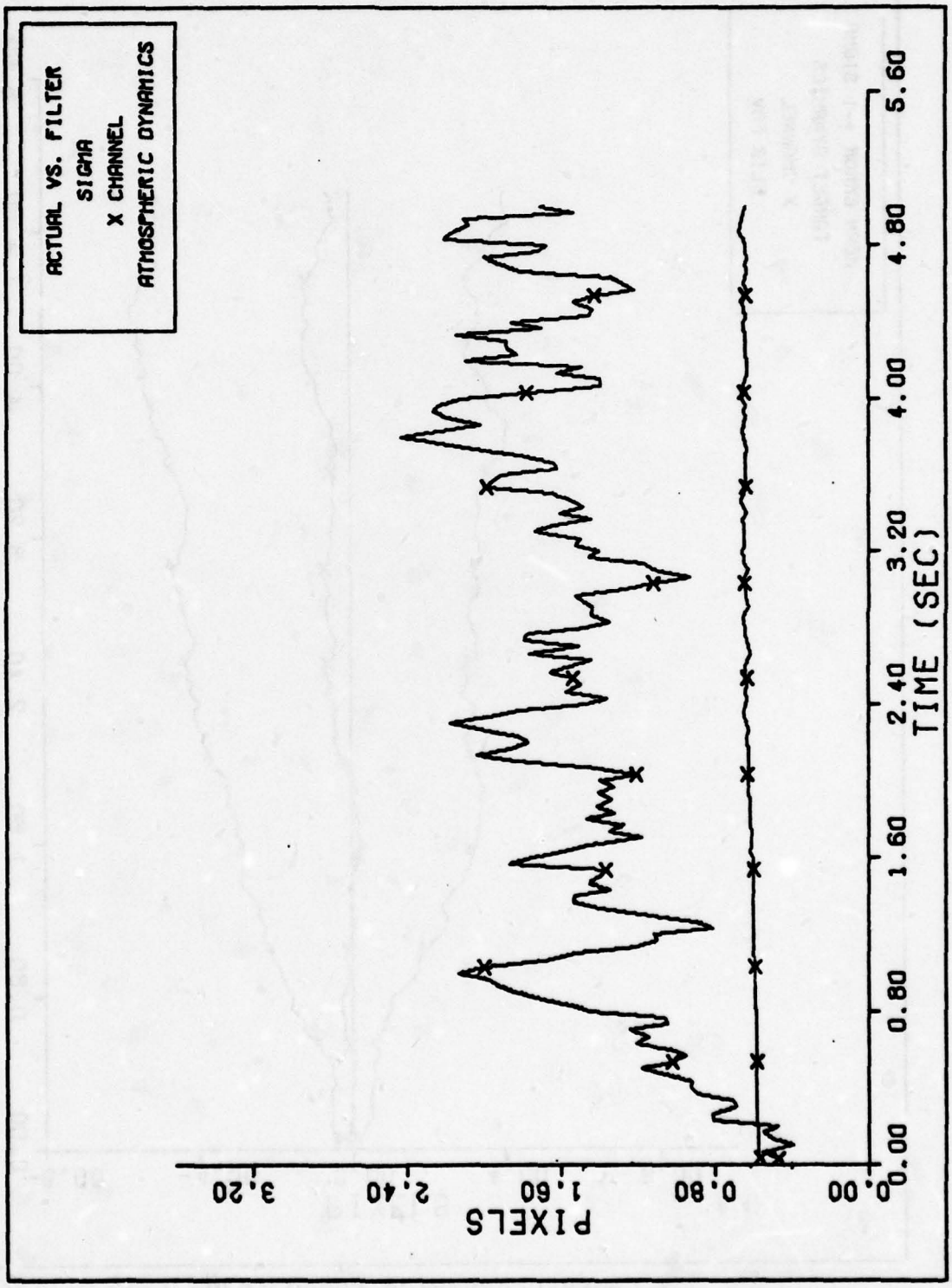
X CHANNEL ATMOSPHERICS ERROR (S/N= 10)

Figure 7a. Case 7 Performance Plot



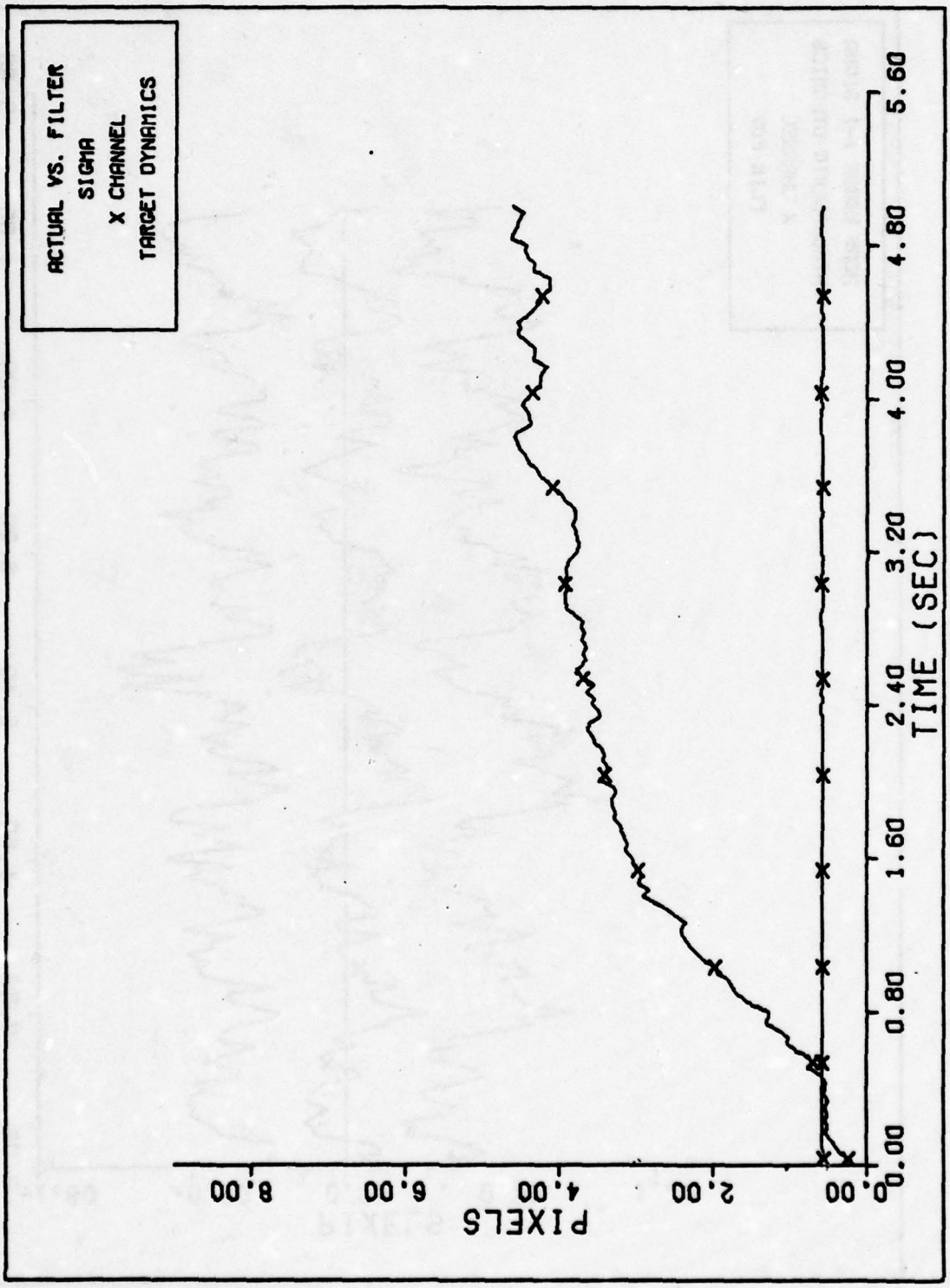
X CHANNEL DYNAMICS ERROR (S/N= 10)

Figure 7b. Case 7 Performance Plot



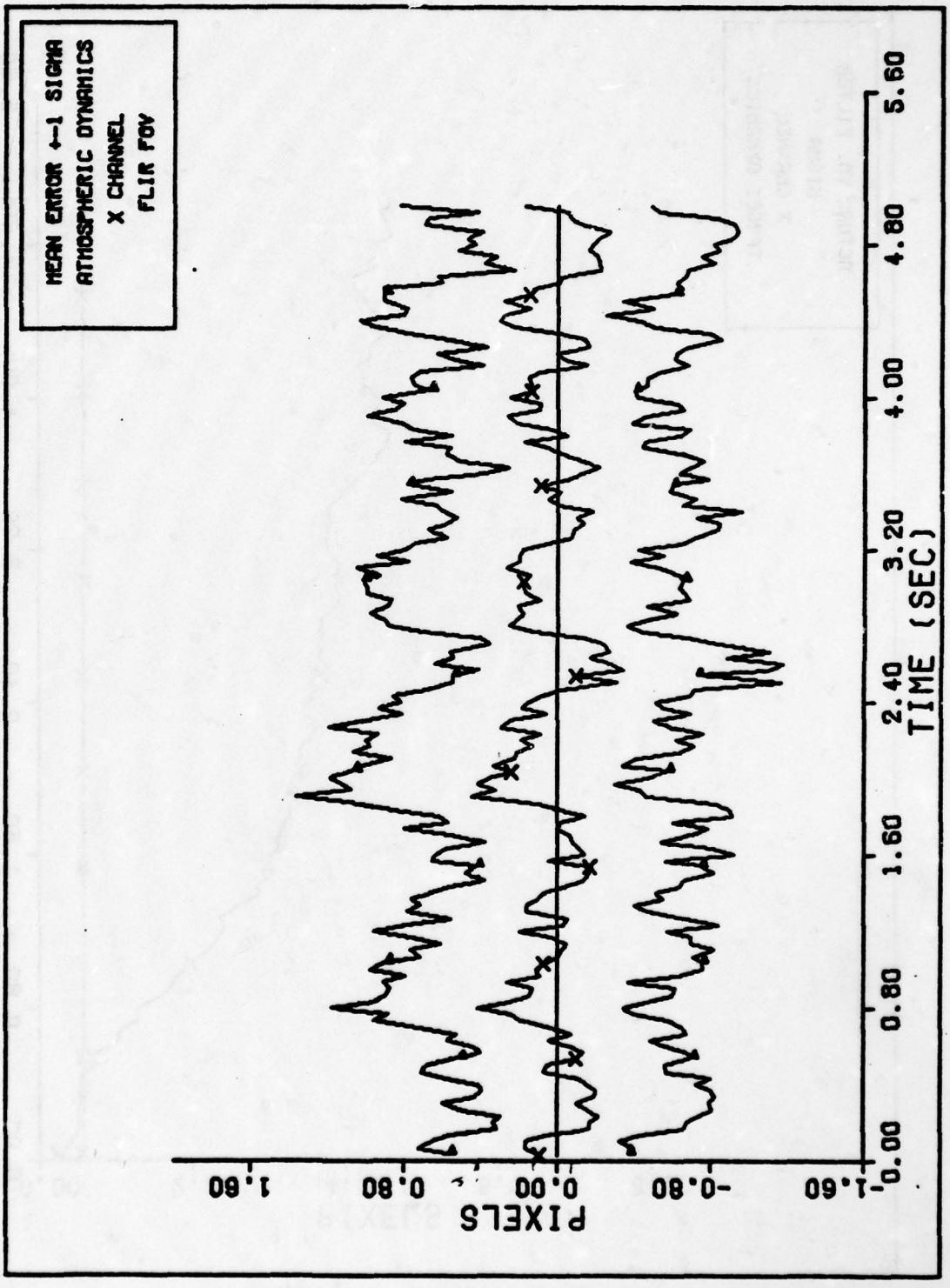
FILTER VS. ACTUAL SIGMA PLOT (S/N = 10)

Figure 7c. Case 7 Performance Plot



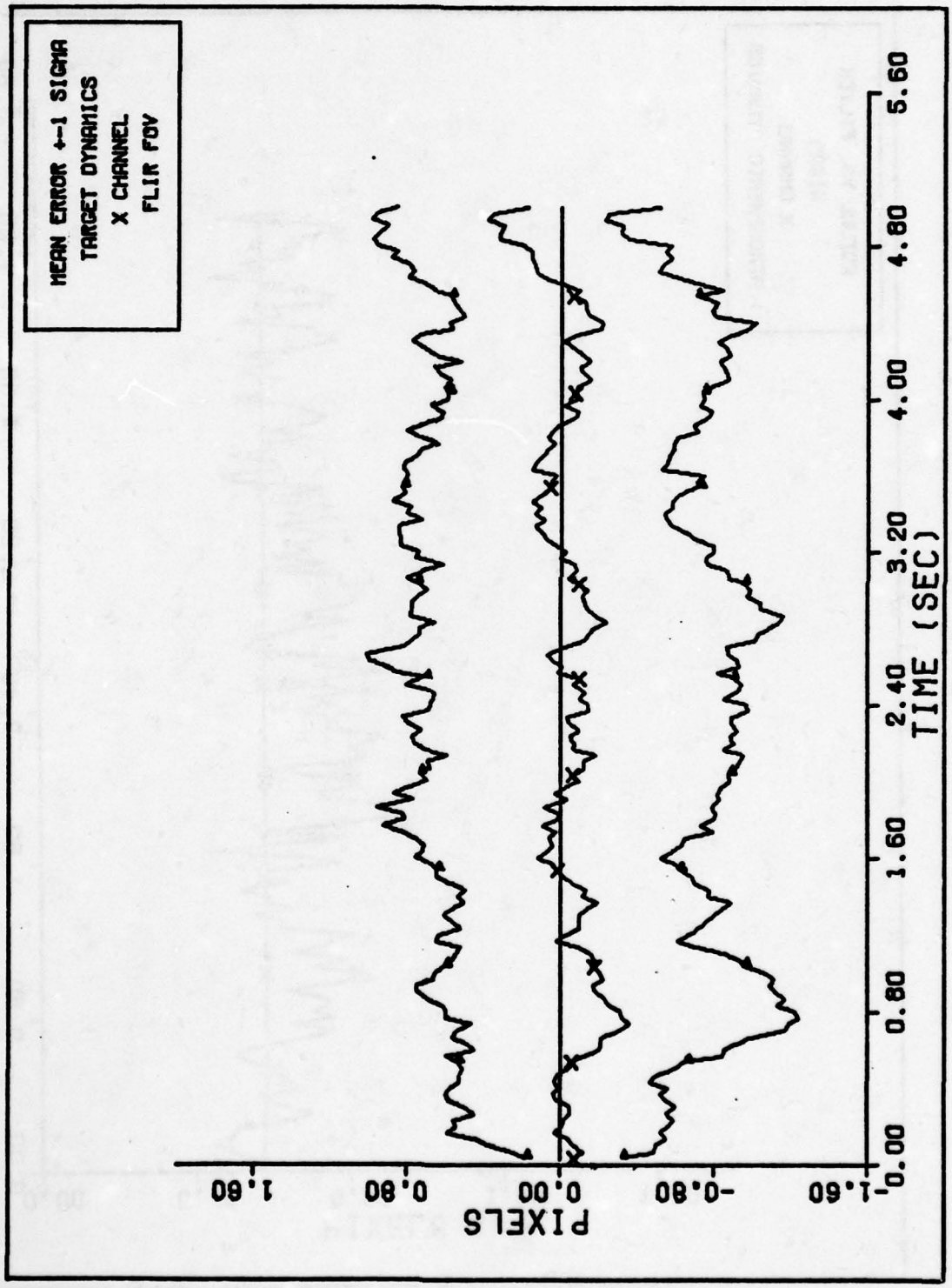
FILTER VS. ACTUAL SIGMA PLOT (S/N = 10)

Figure 7d. Case 7 Performance Plot



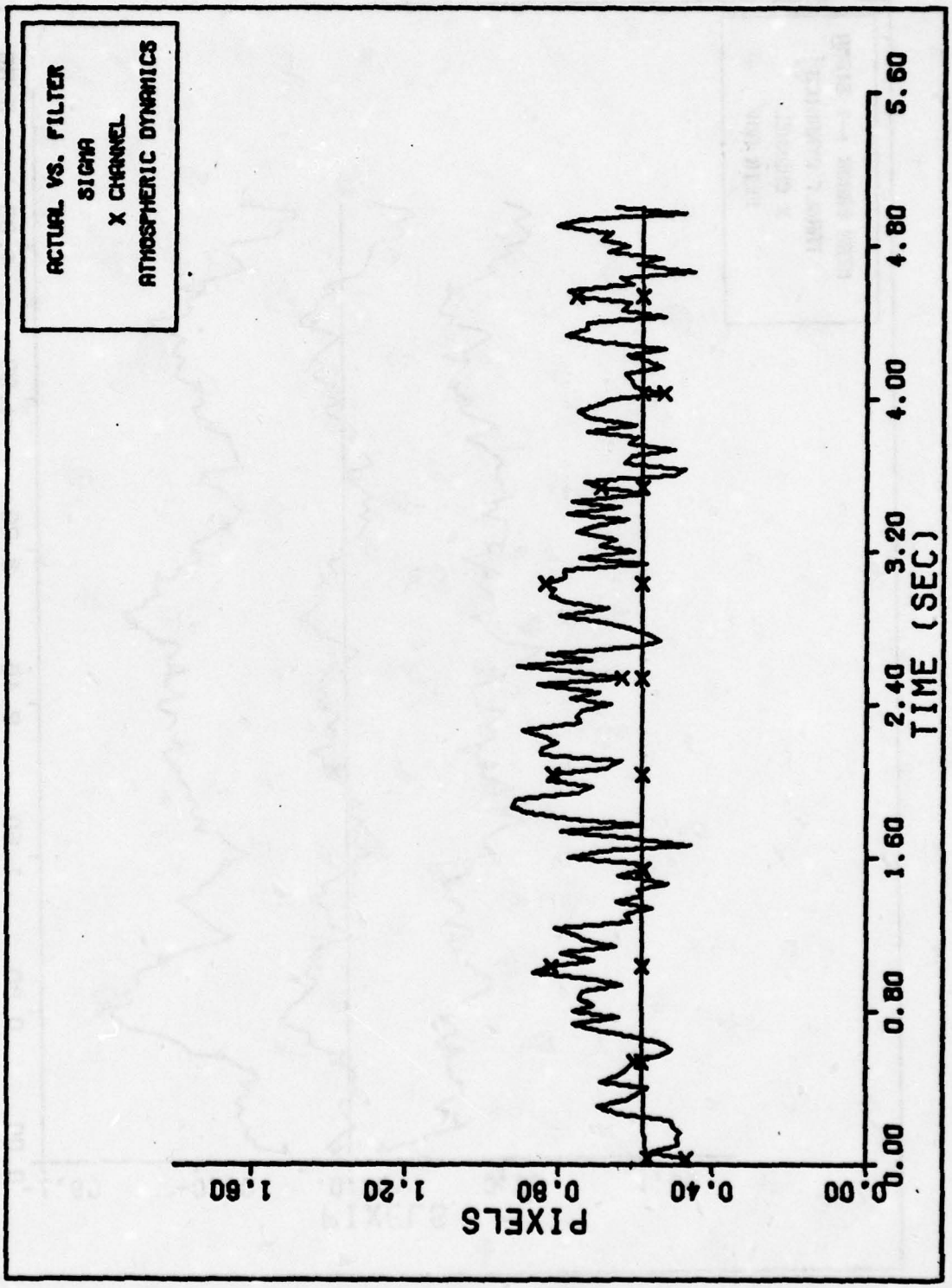
X CHANNEL ATMOSPHERICS ERROR (S/N= 10)

Figure 8a. Case 8 Performance Plot



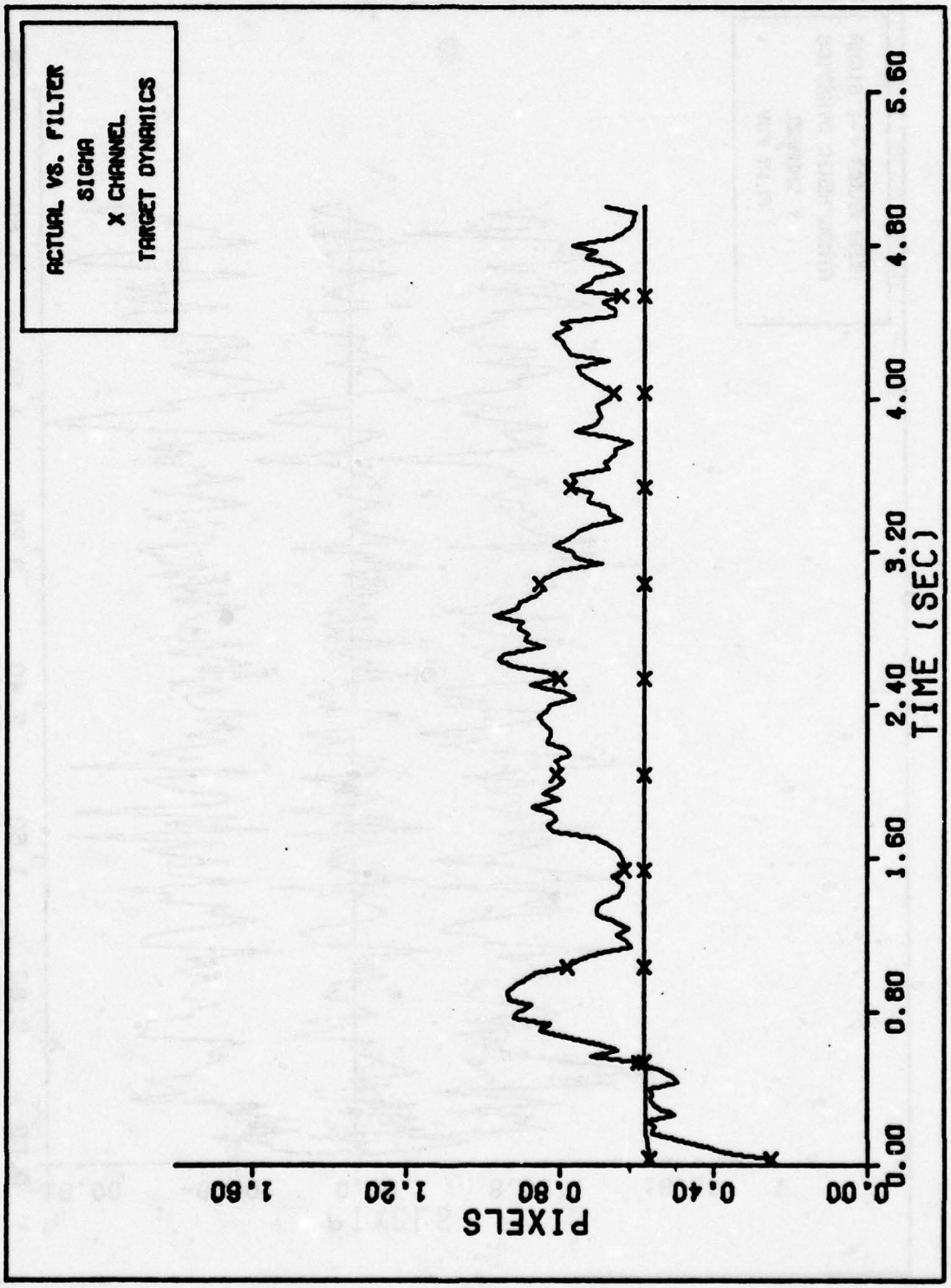
X CHANNEL DYNAMICS ERROR (S/N= 10)

Figure 8b. Case 8 Performance Plot



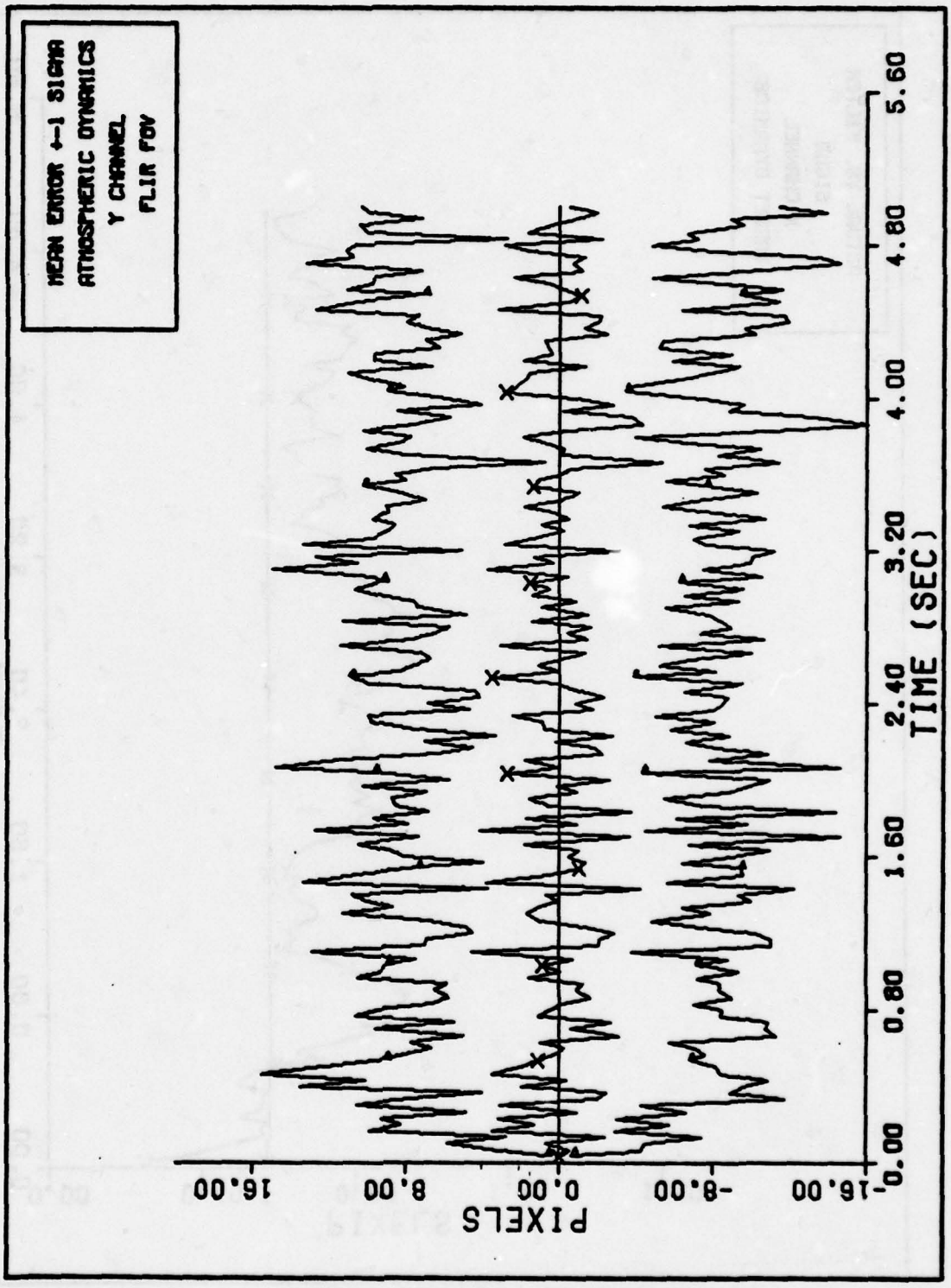
FILTER VS. ACTUAL SIGMA PLOT (S/N = 10)

Figure 8c. Case 8 Performance Plot



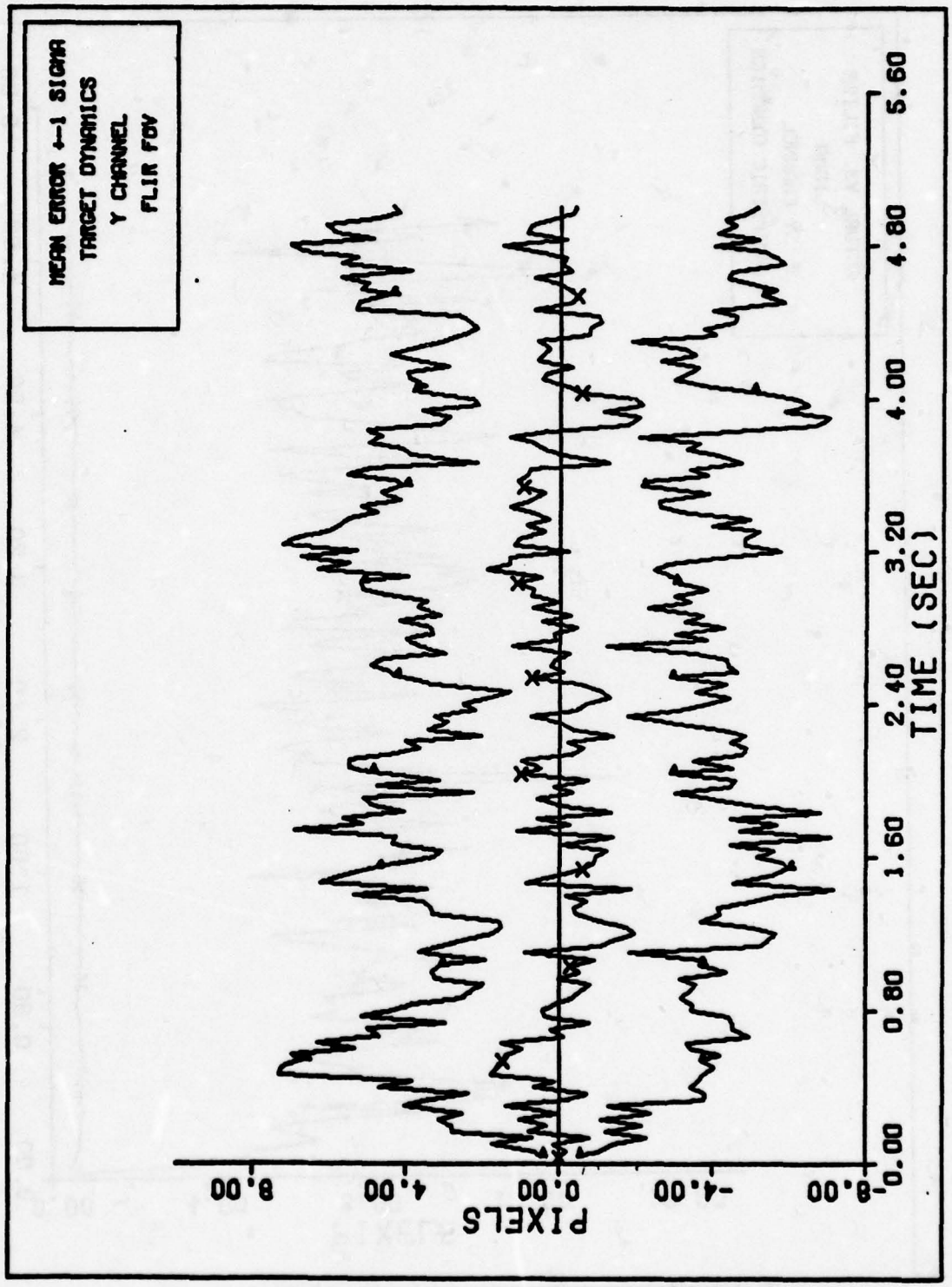
FILTER VS. ACTUAL SIGMA PLOT (S/N = 10)

Figure 8d. Case 8 Performance Plot



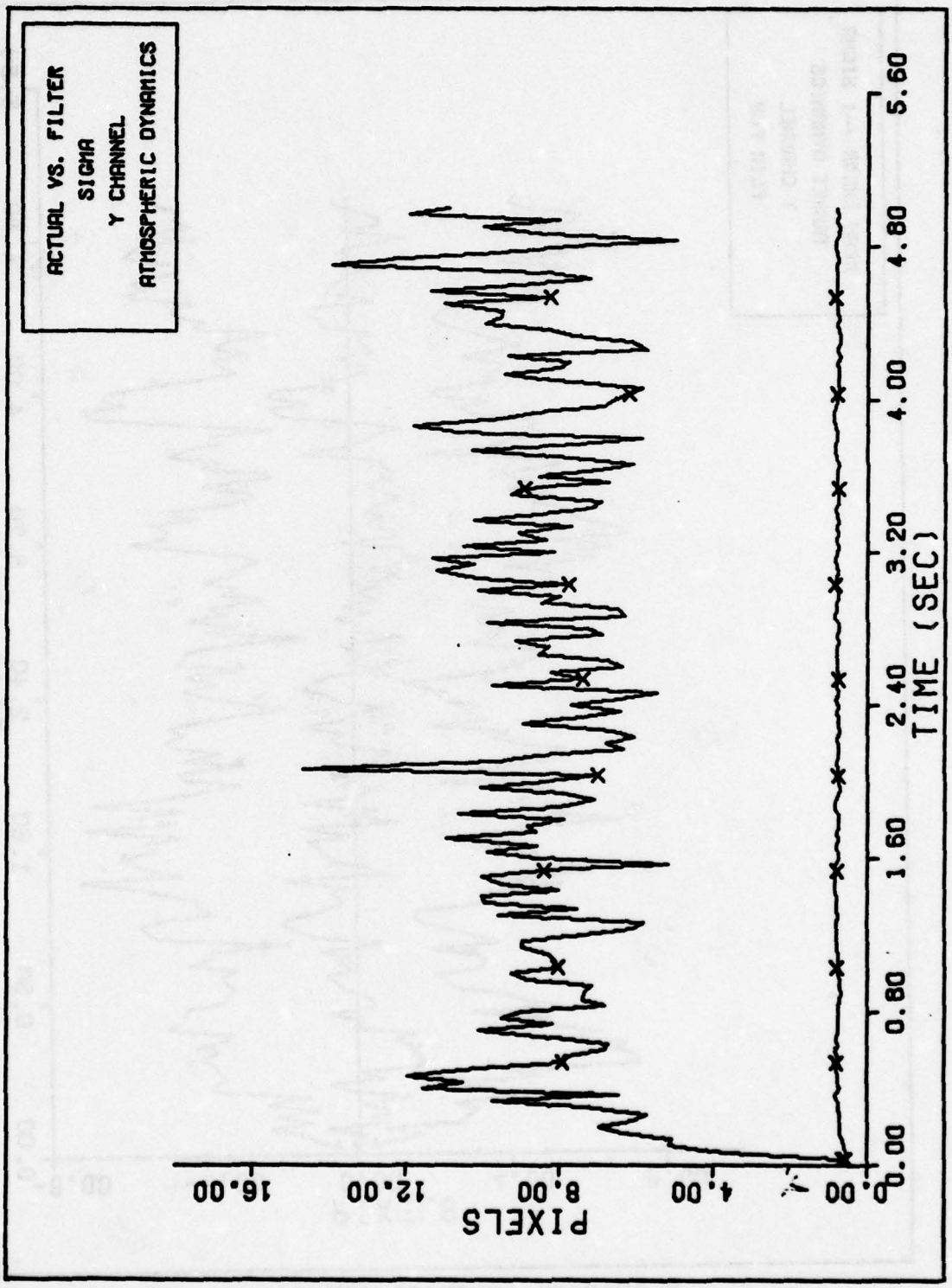
Y CHANNEL ATMOSPHERICS ERROR (S/N = 1)

Figure 9a. Case 9 Performance Plot



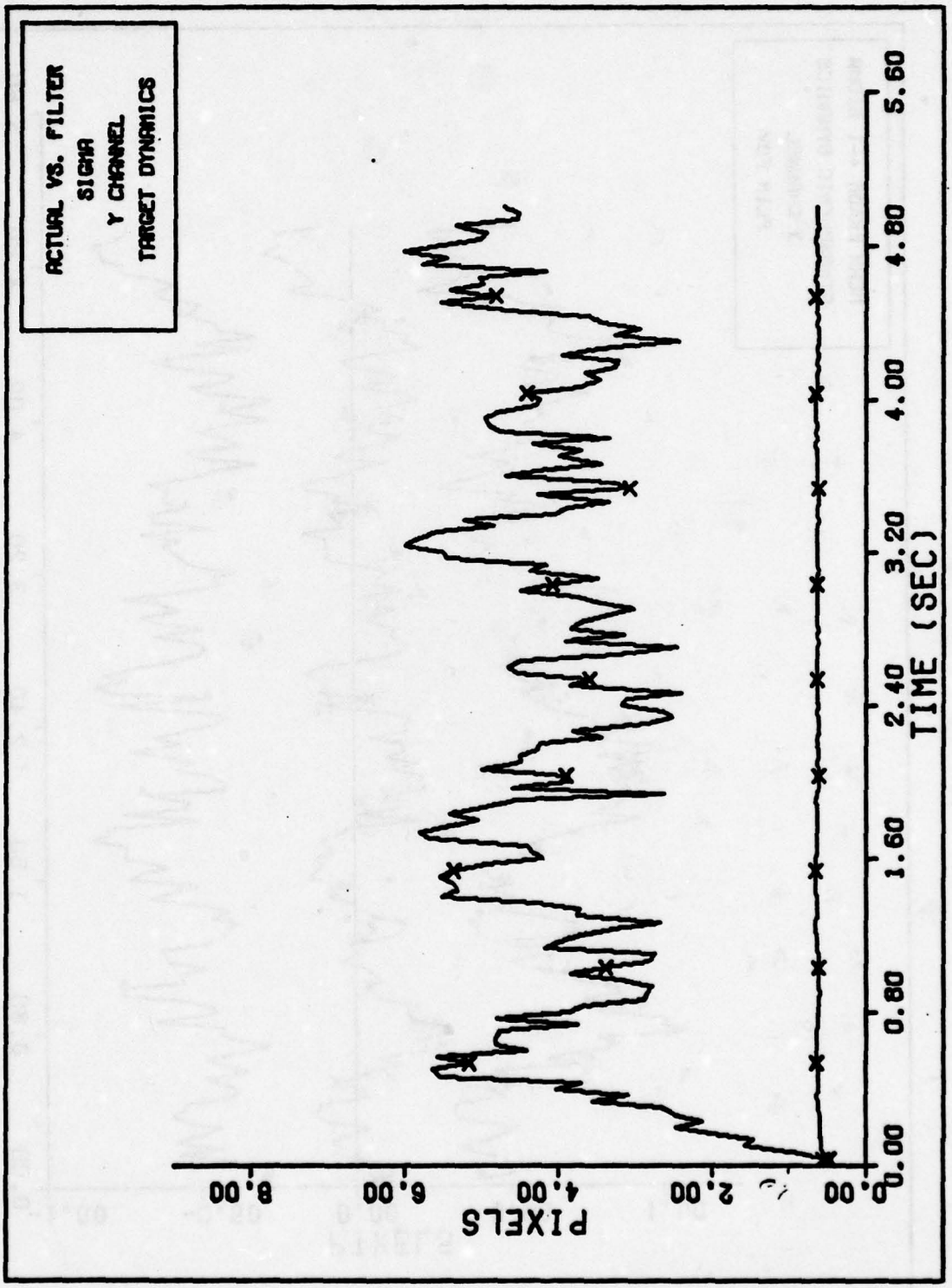
Y CHANNEL DYNAMICS ERROR (S/N= 1)

Figure 9b. Case 9 Performance Plot



FILTER VS. ACTUAL SIGMA PLOT (S/N = 1)

Figure 9c. Case 9 Performance Plot



FILTER VS. ACTUAL SIGMA PLOT (S/N = 1)

Figure 9d. Case 9 Performance Plot

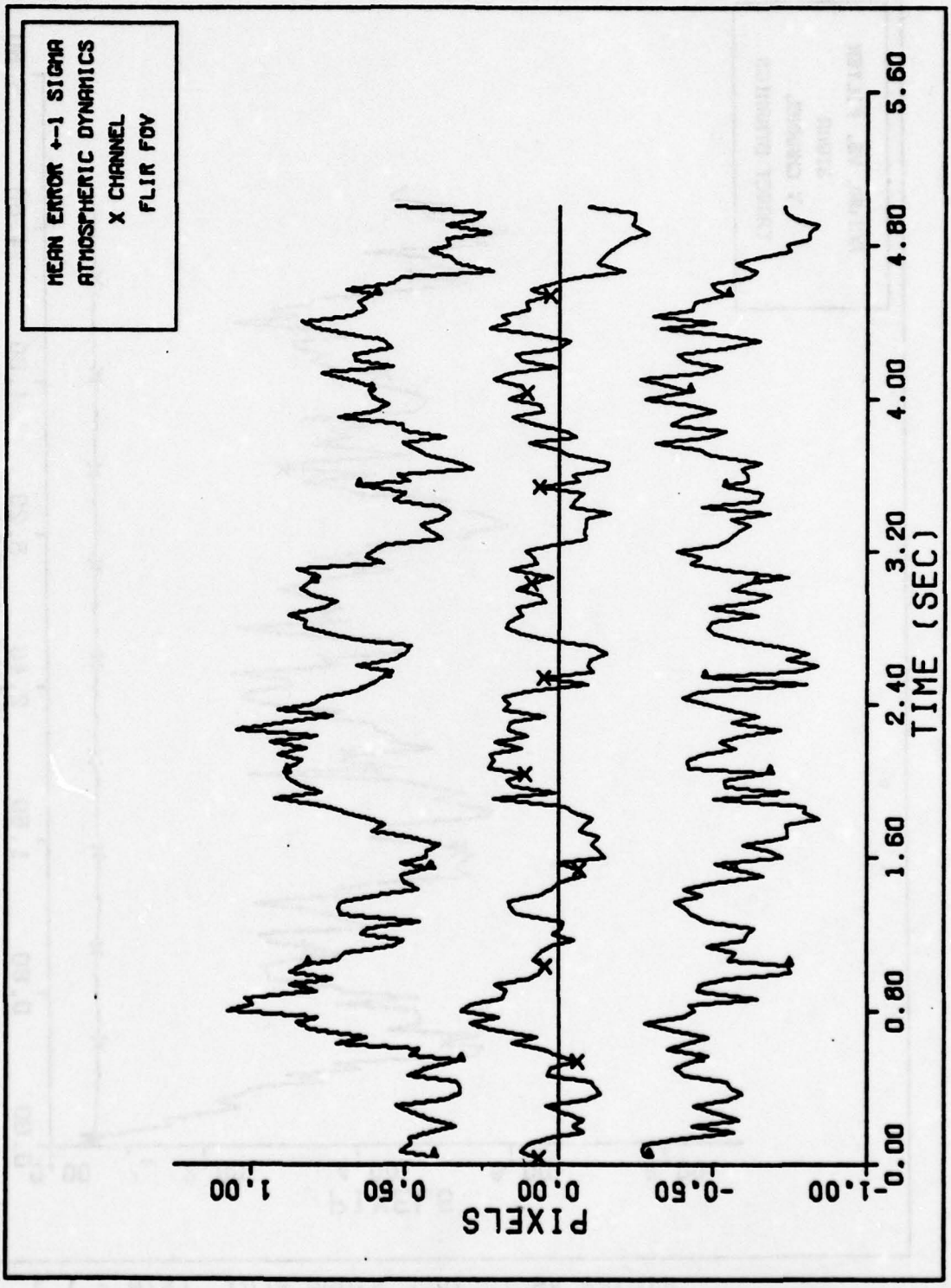
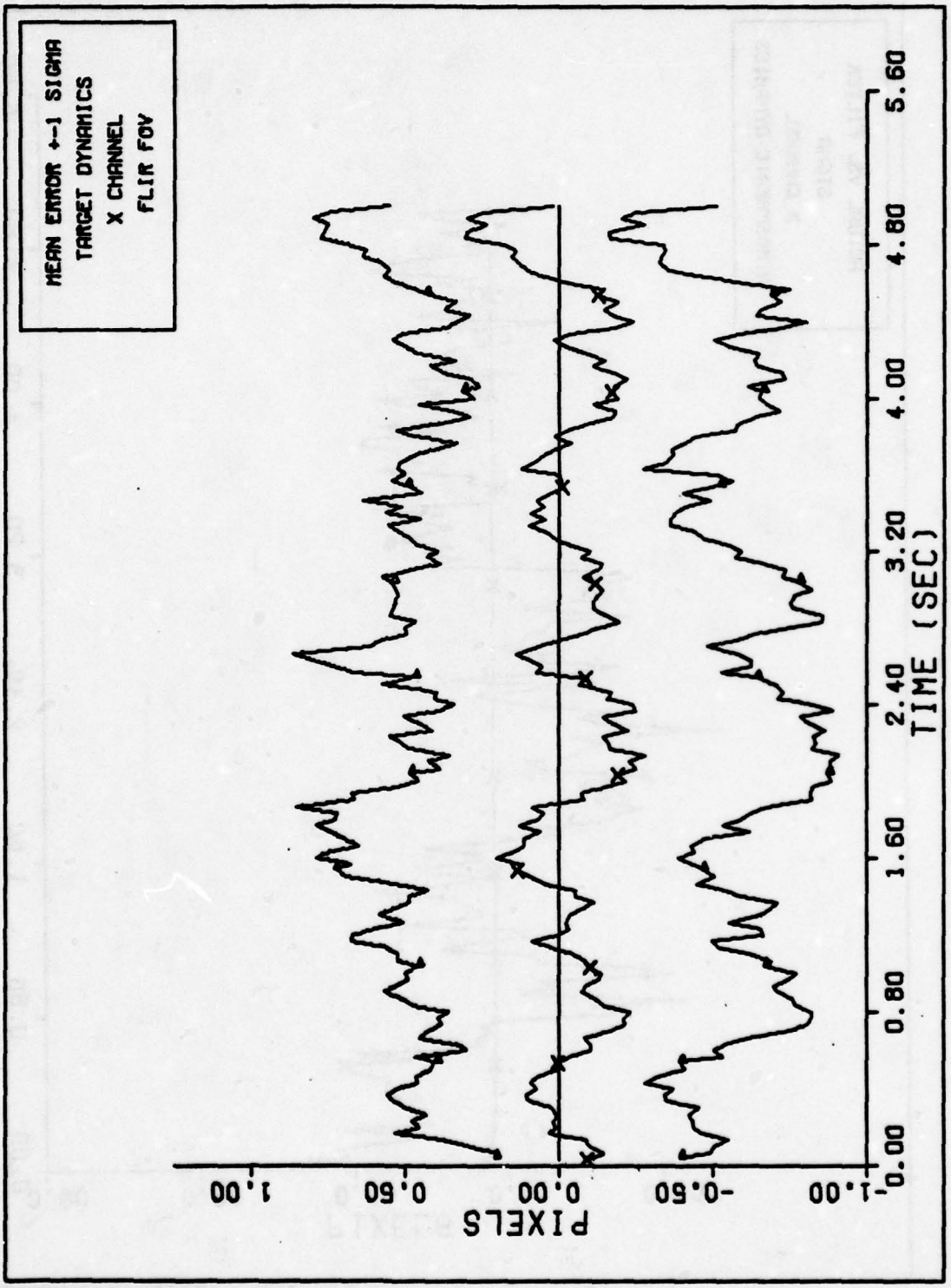
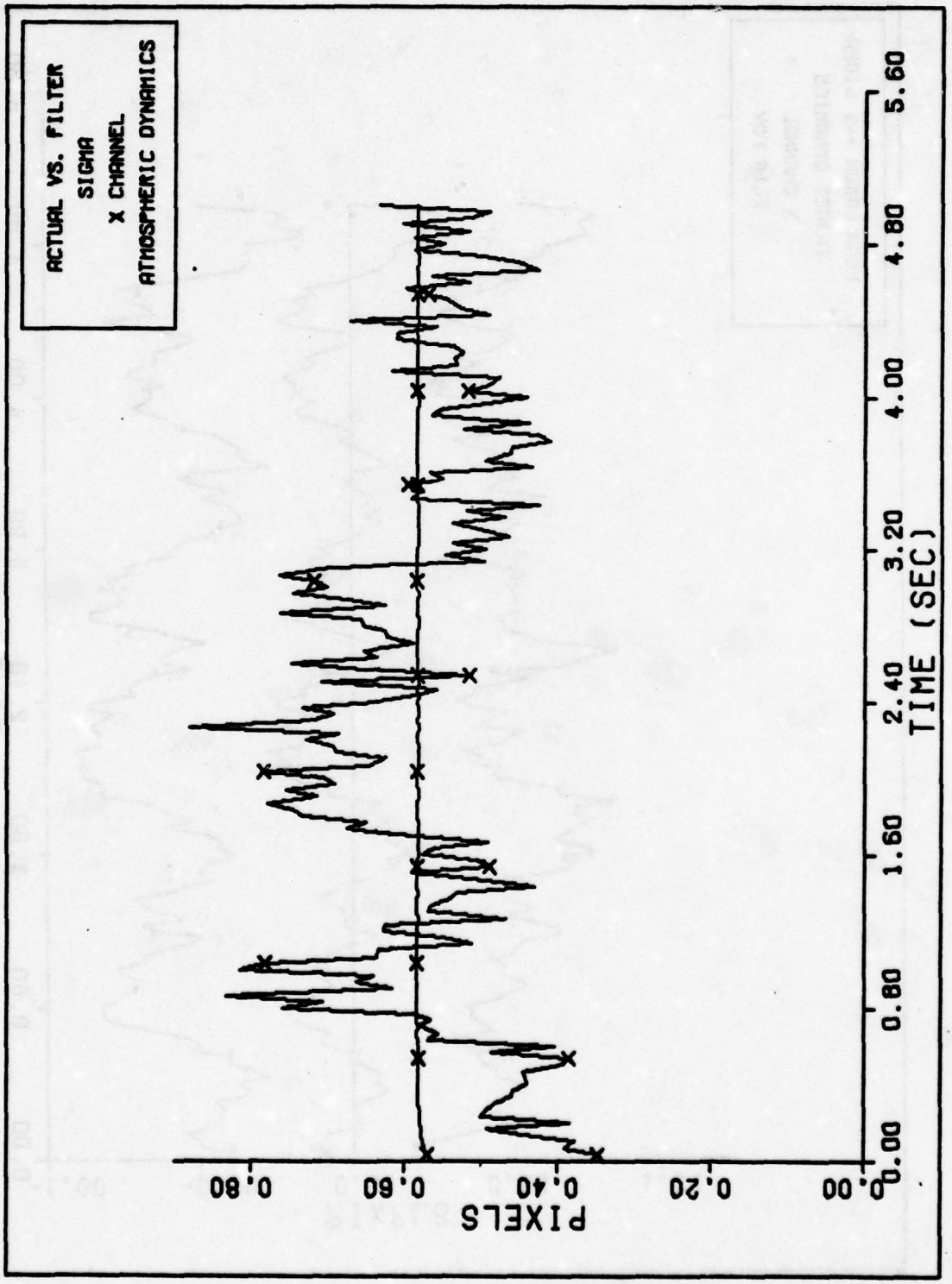


Figure 10a. Case 10 Performance Plot



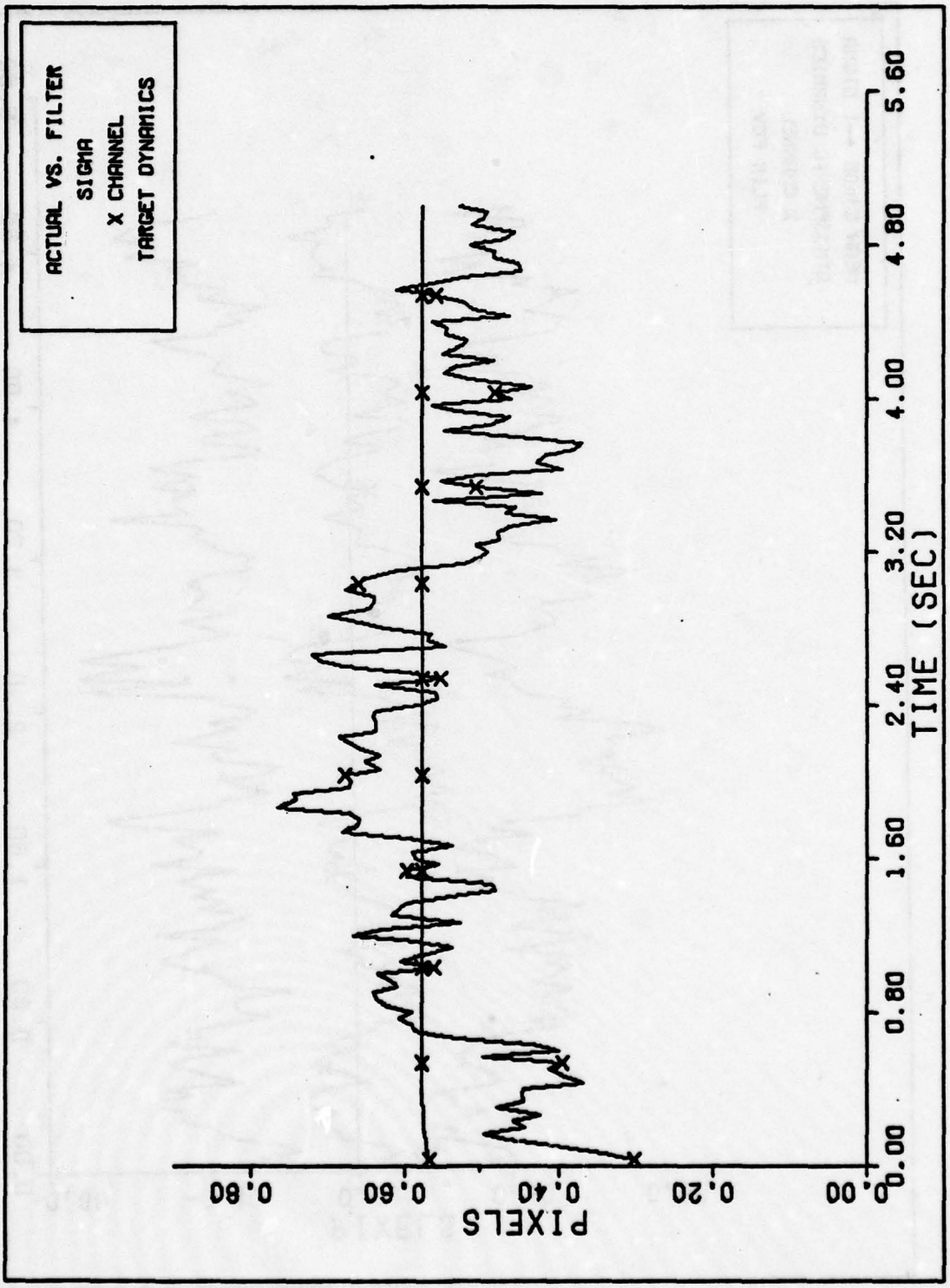
X CHANNEL DYNAMICS ERROR

Figure 10b. Case 10 Performance Plot



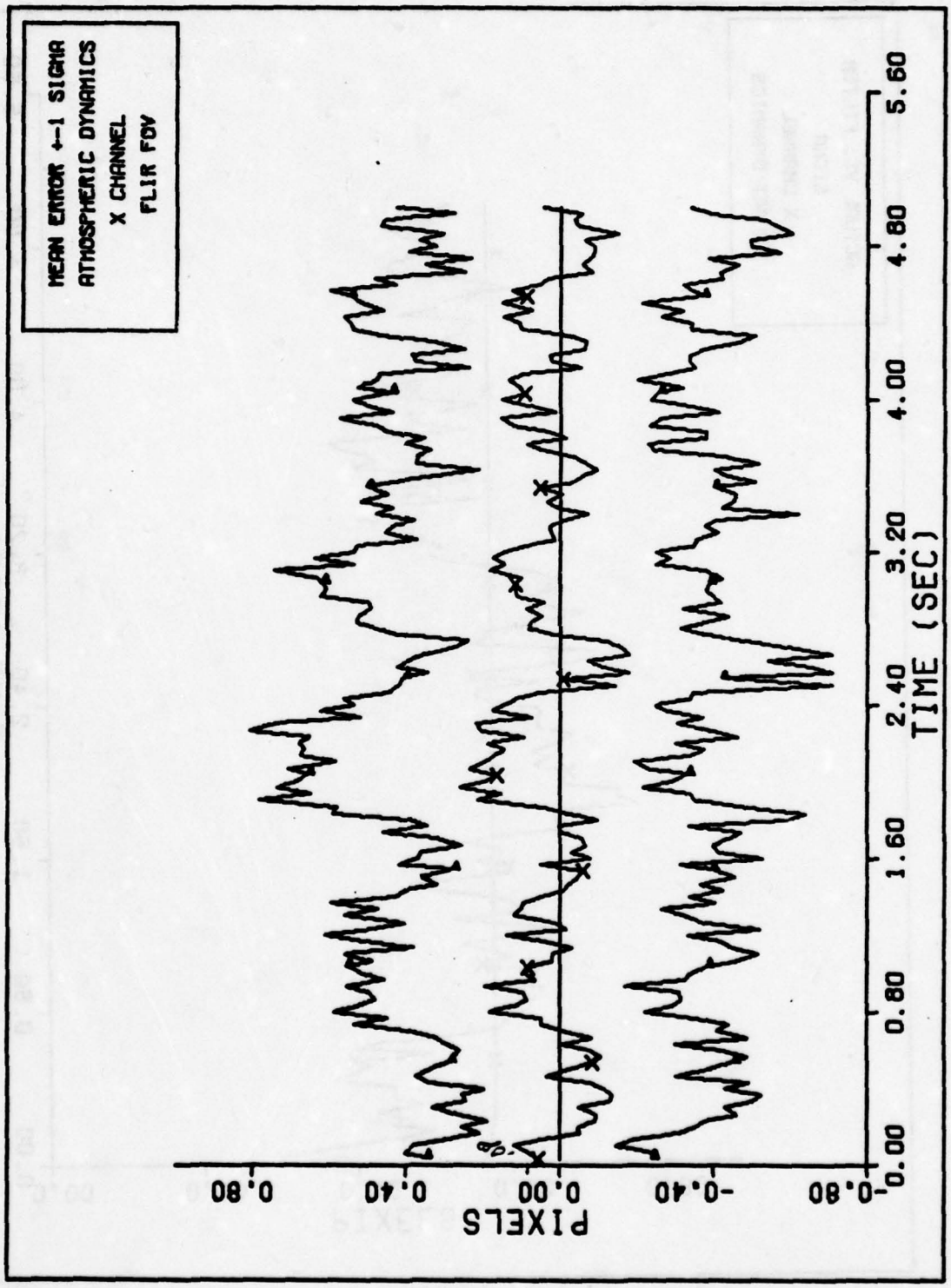
FILTER VS. ACTUAL SIGMA PLOT

Figure 10c. Case 10 Performance Plot



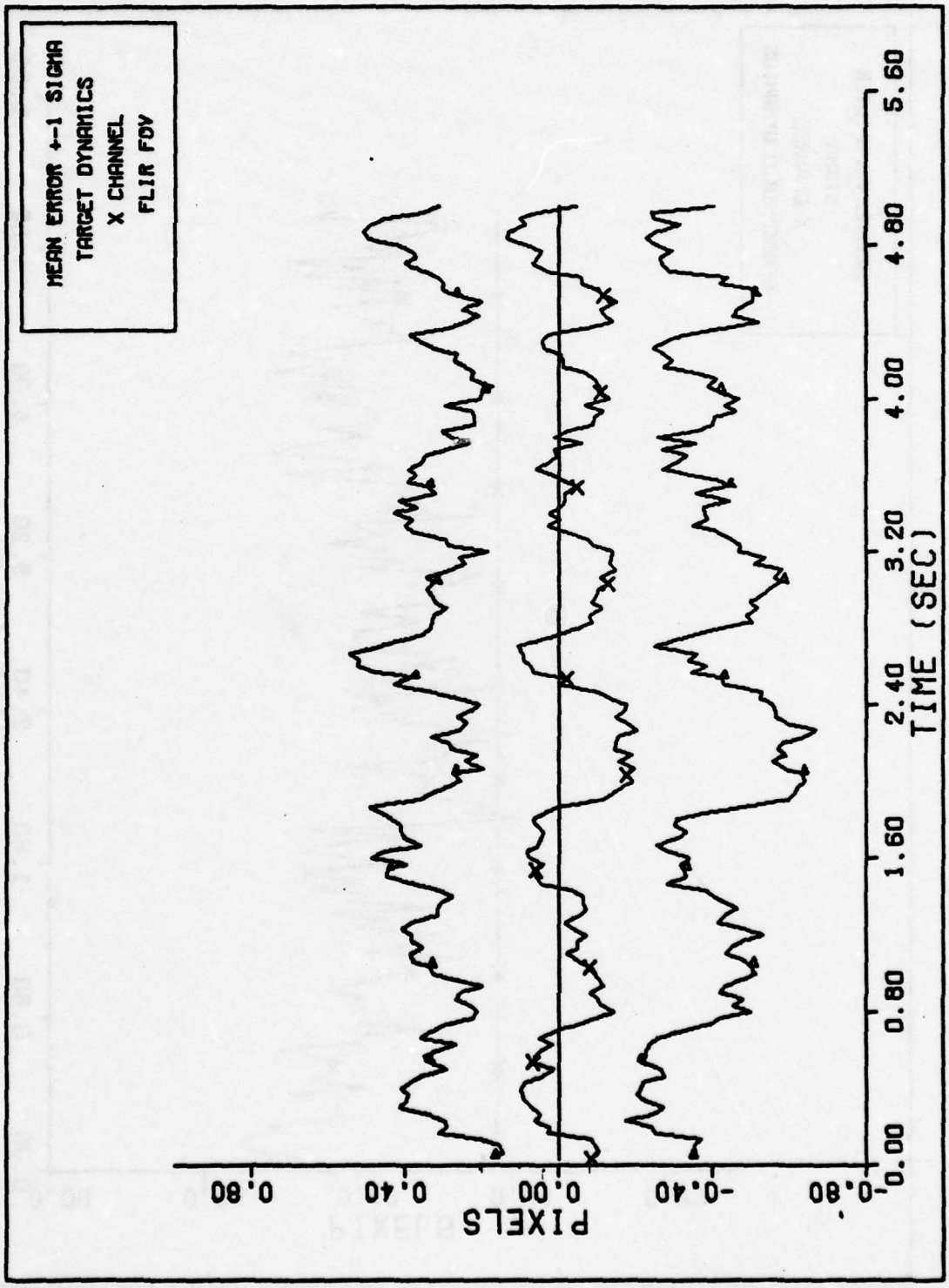
FILTER VS. ACTUAL SIGMA PLOT

Figure 10d. Case 10 Performance Plot



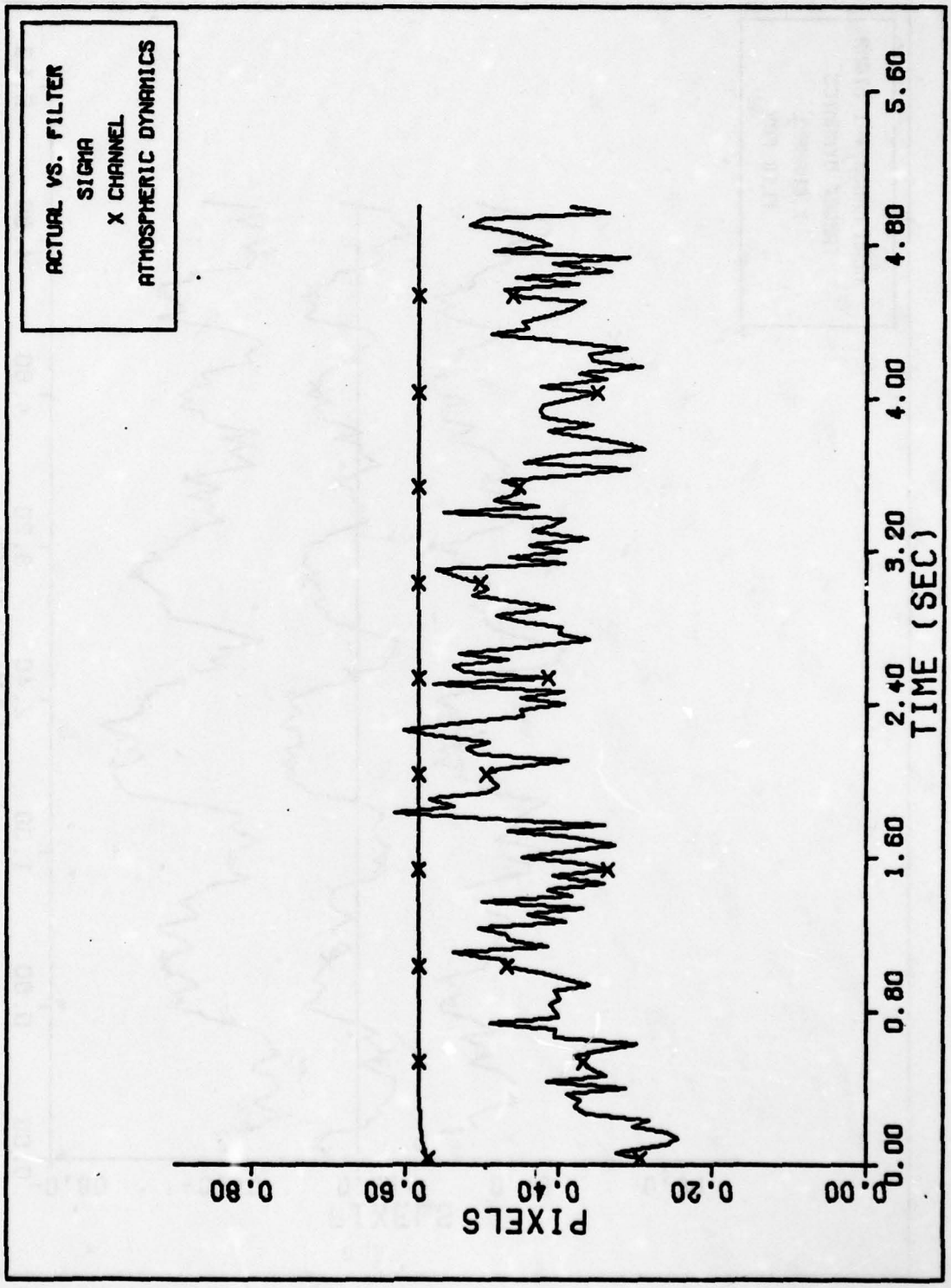
X CHANNEL ATMOSPHERICS ERROR (S/N= 10)

Figure 11a. Case 11 Performance Plot



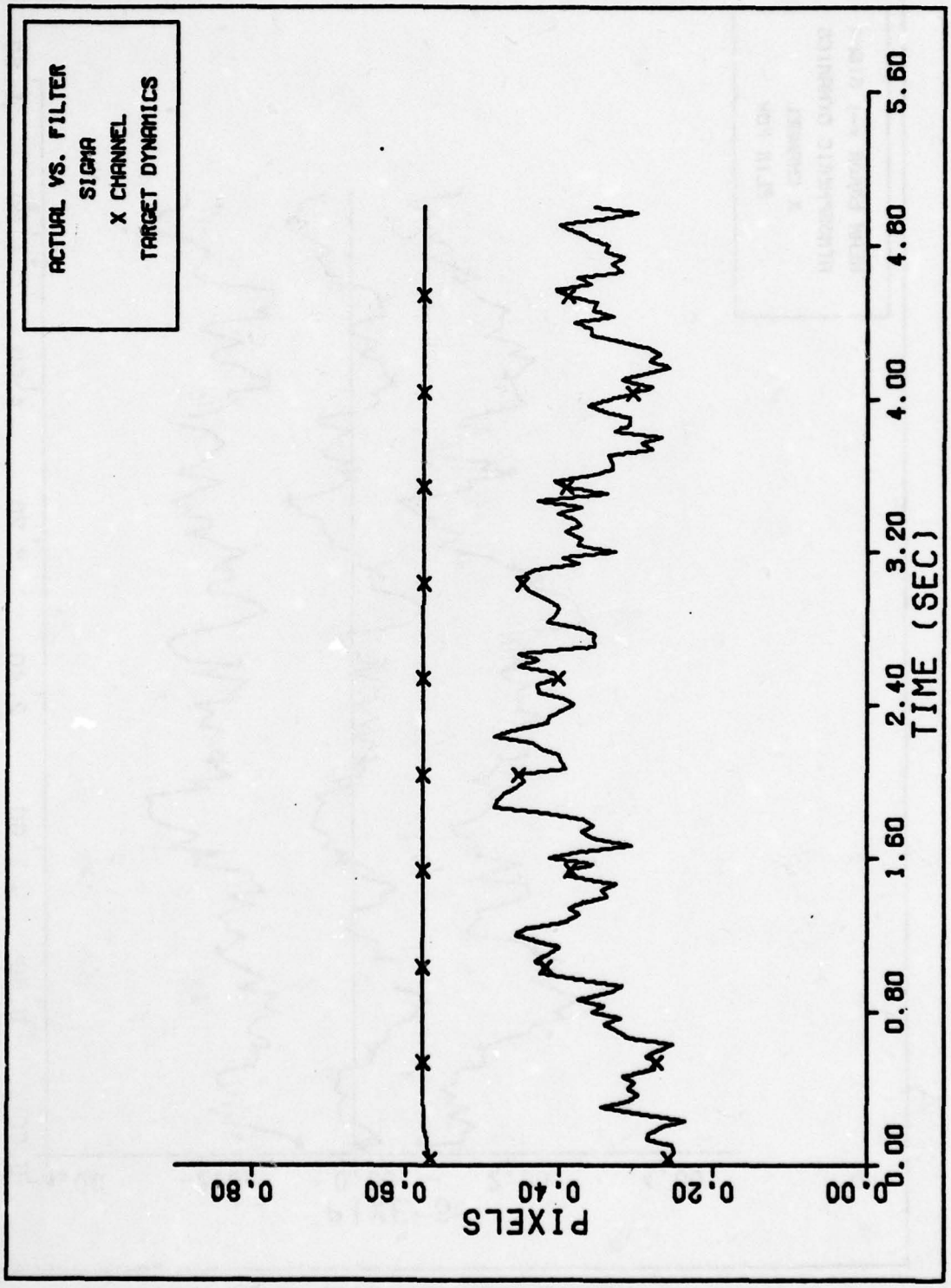
X CHANNEL DYNAMICS ERROR (S/N= 10)

Figure 11b. Case 11 Performance Plot



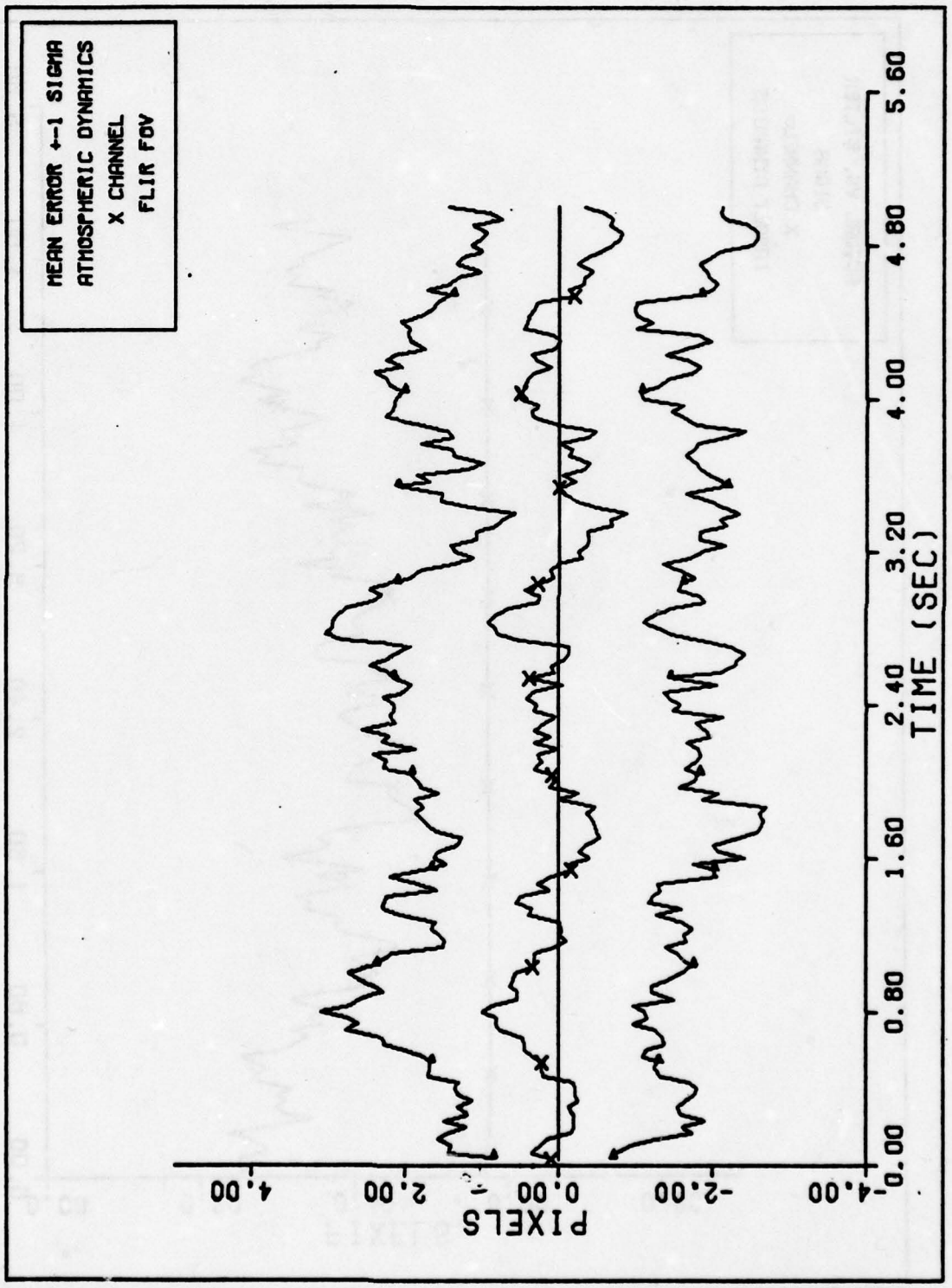
FILTER VS. ACTUAL SIGMA PLOT (S/N = 10)

Figure 11c. Case 11 Performance Plot



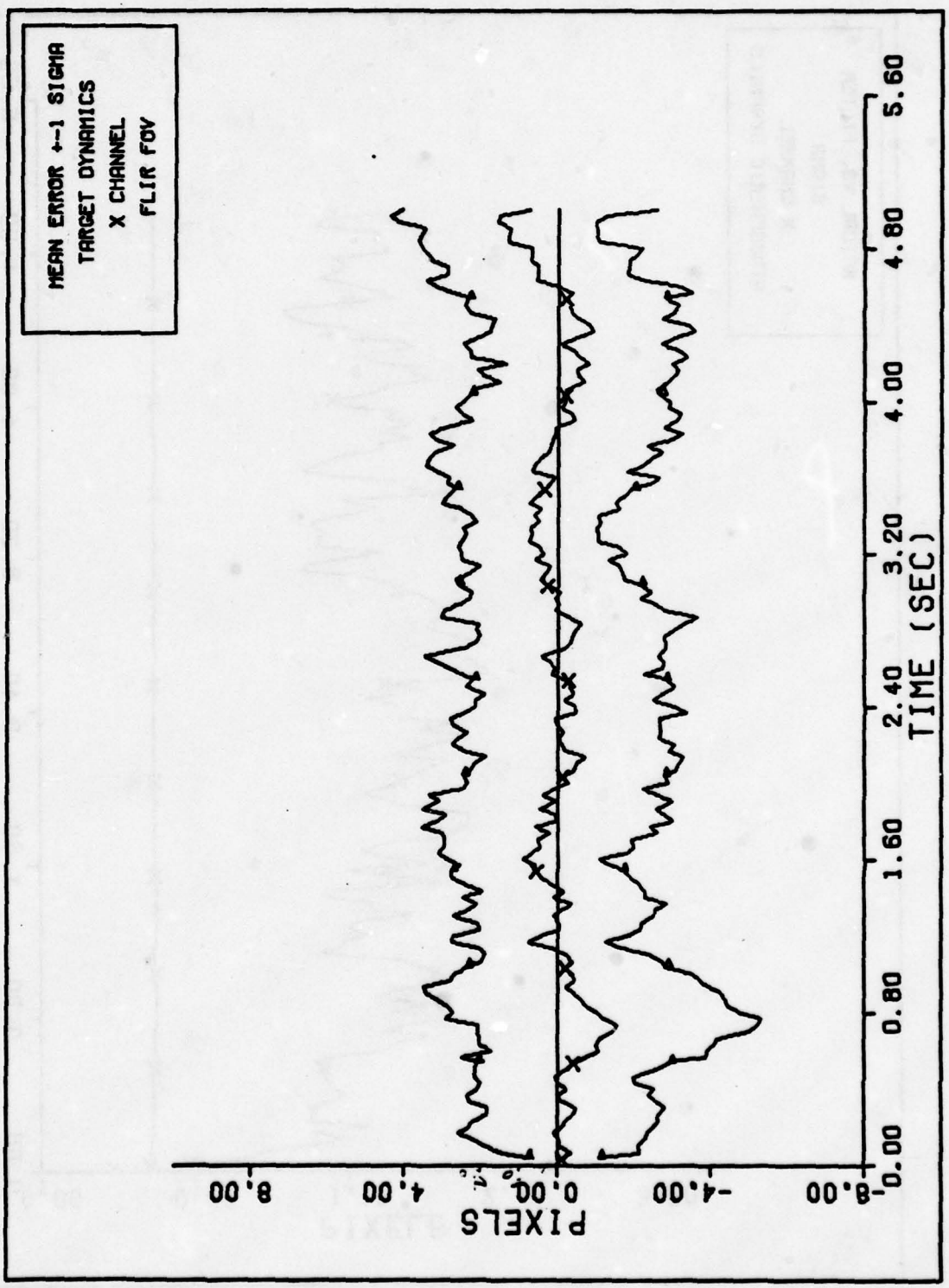
FILTER VS. ACTUAL SIGMA PLOT (S/N = 10)

Figure 11d. Case 11 Performance Plot



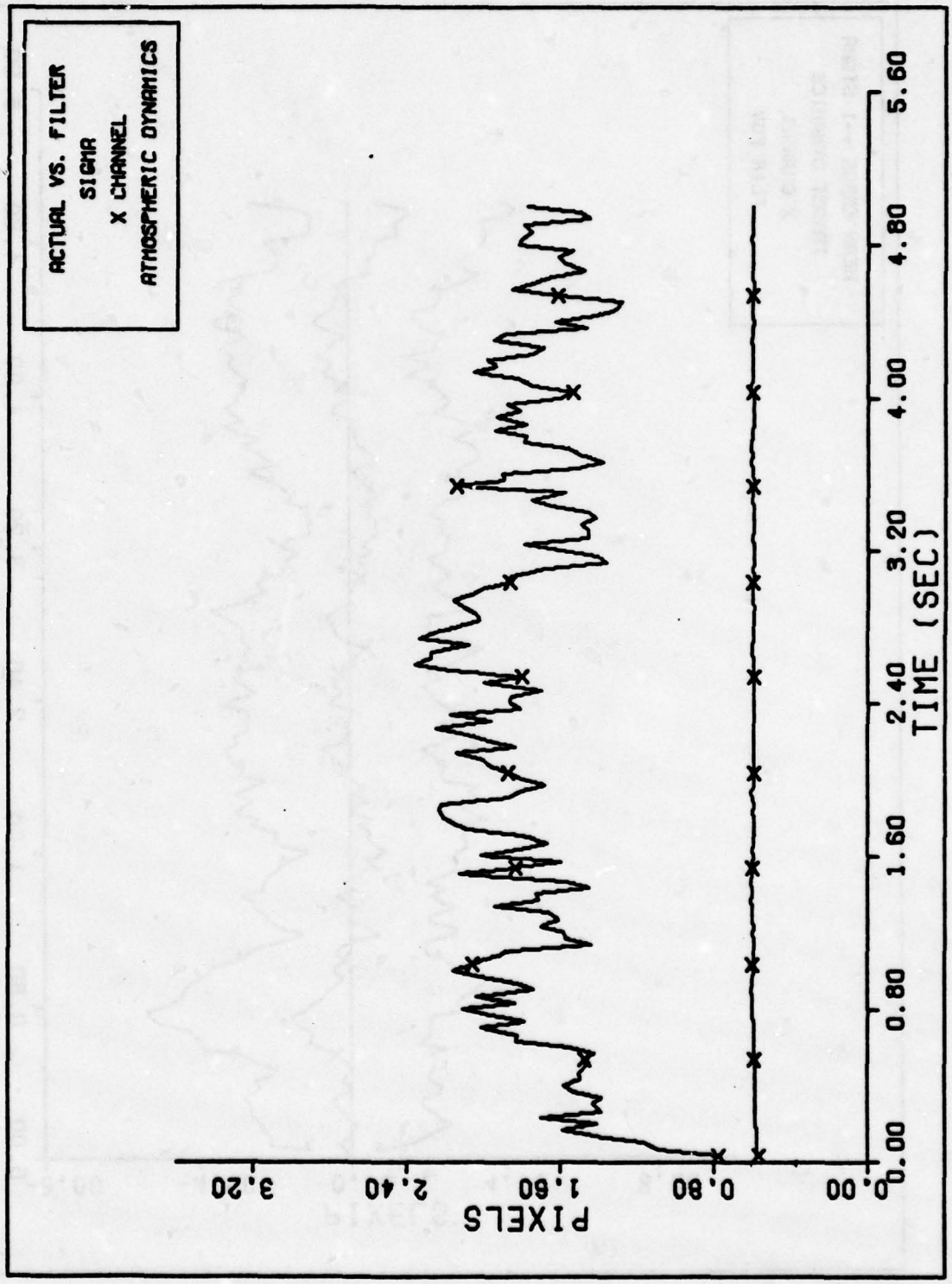
X CHANNEL ATMOSPHERICS ERROR (S/N= 10)

Figure 12a. Case 12 Performance Plot



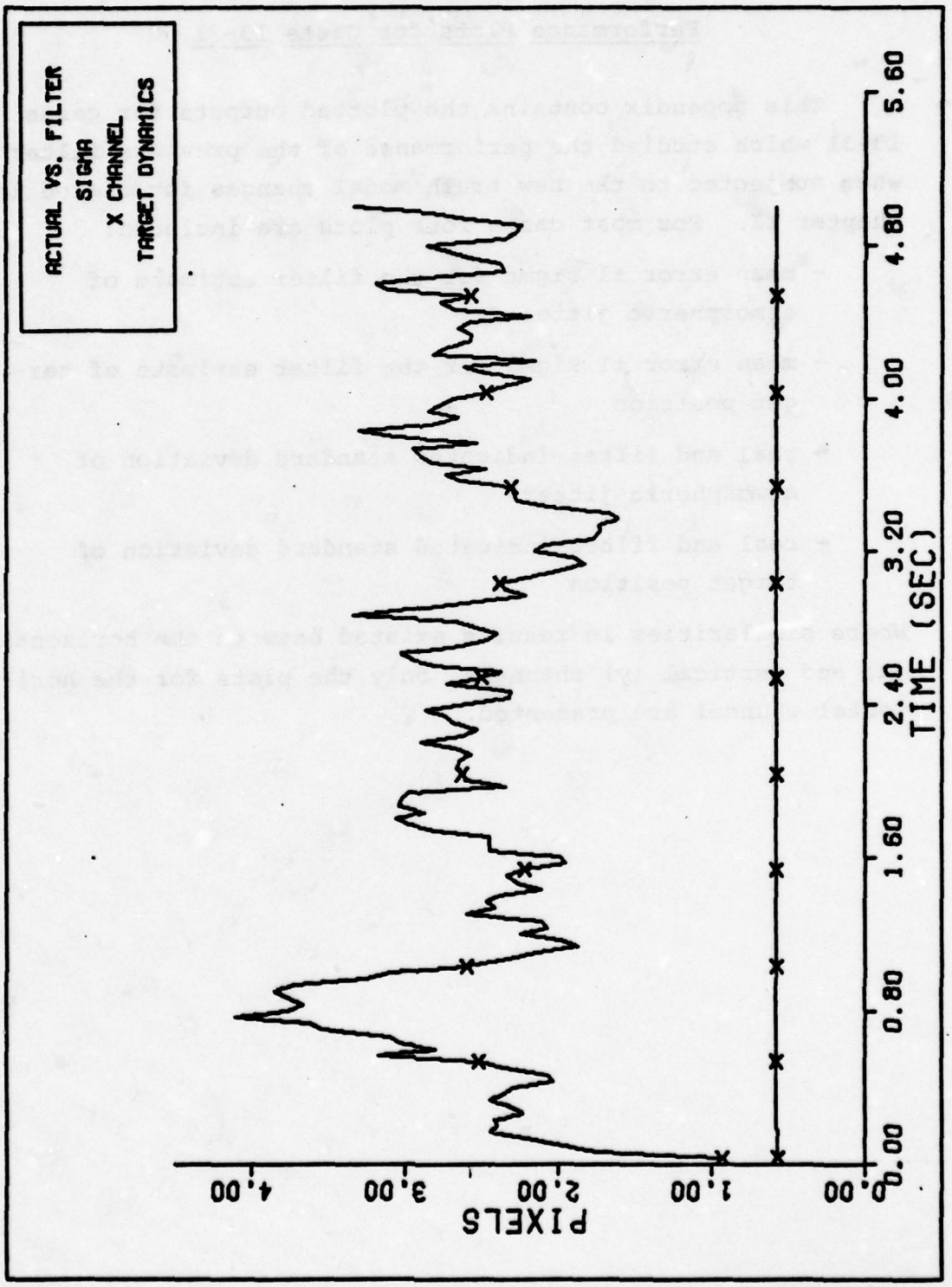
X CHANNEL DYNAMICS ERROR (S/N= 10)

Figure 12b. Case 12 Performance Plot



FILTER VS. ACTUAL SIGMA PLOT (S/N = 10)

Figure 12c. Case 12 Performance Plot



FILTER VS. ACTUAL SIGMA PLOT (S/N = 10)

Figure 12d. Case 12 Performance Plot

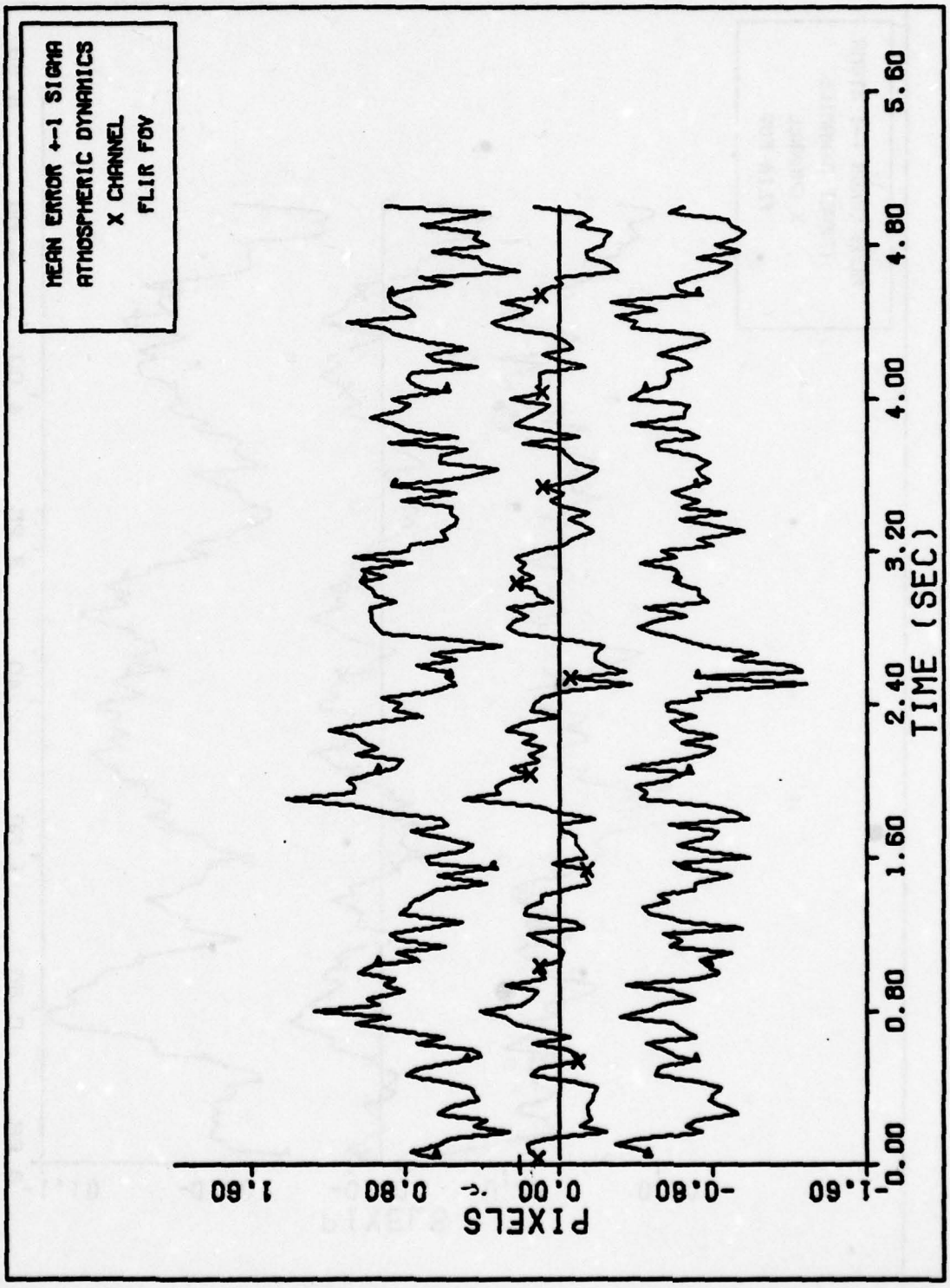
Appendix J

Performance Plots for Cases 13-31

This appendix contains the plotted outputs for cases 13-31 which studied the performance of the previous filter when subjected to the new truth model changes formulated in Chapter II. For most cases four plots are included:

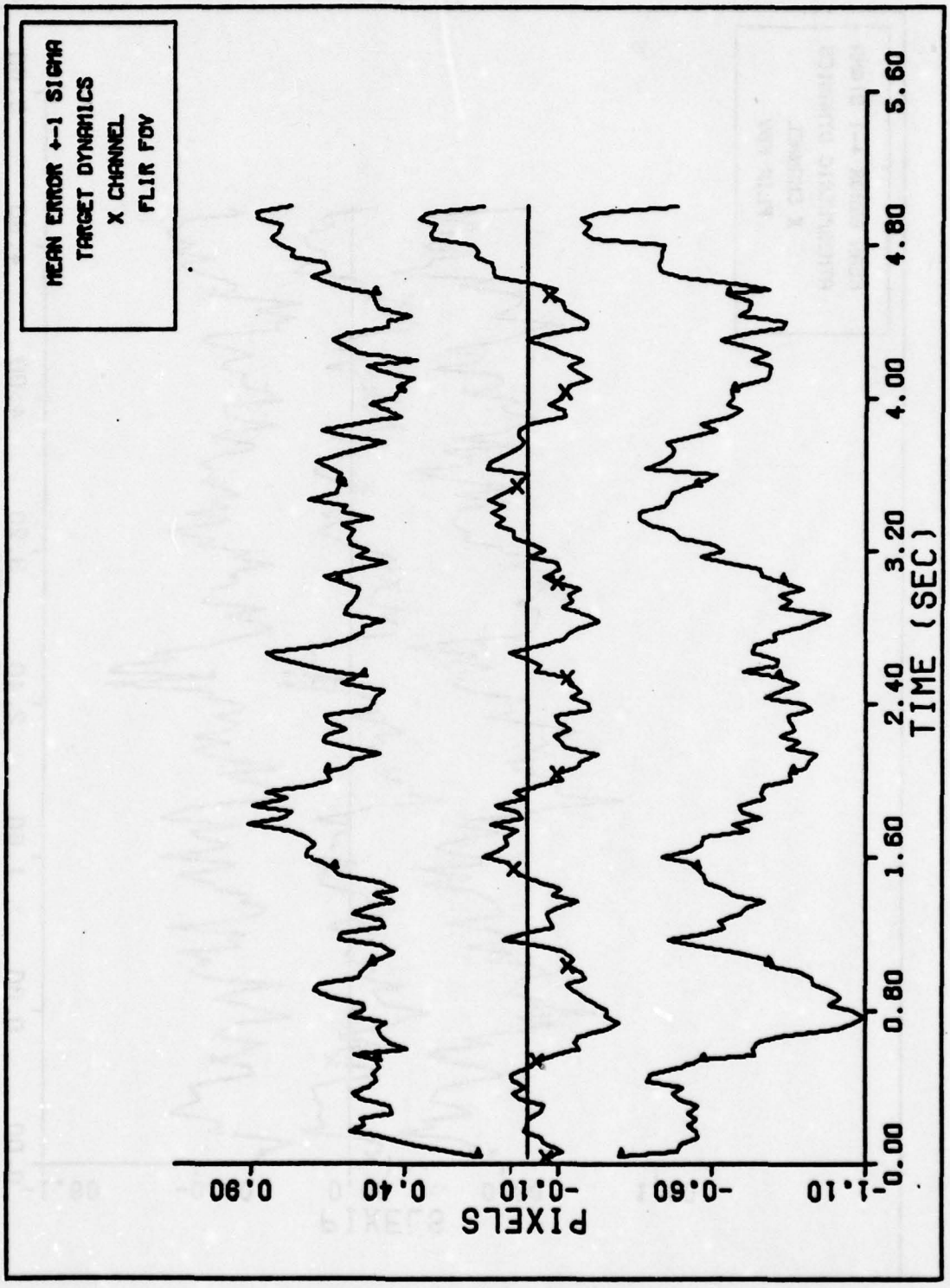
- mean error ±1 sigma for the filter estimate of atmospheric jitter
- mean error ±1 sigma for the filter estimate of target position
- real and filter-indicated standard deviation of atmospheric jitter
- real and filter-indicated standard deviation of target position

Where similarities in results existed between the horizontal (x) and vertical (y) channels, only the plots for the horizontal channel are presented.



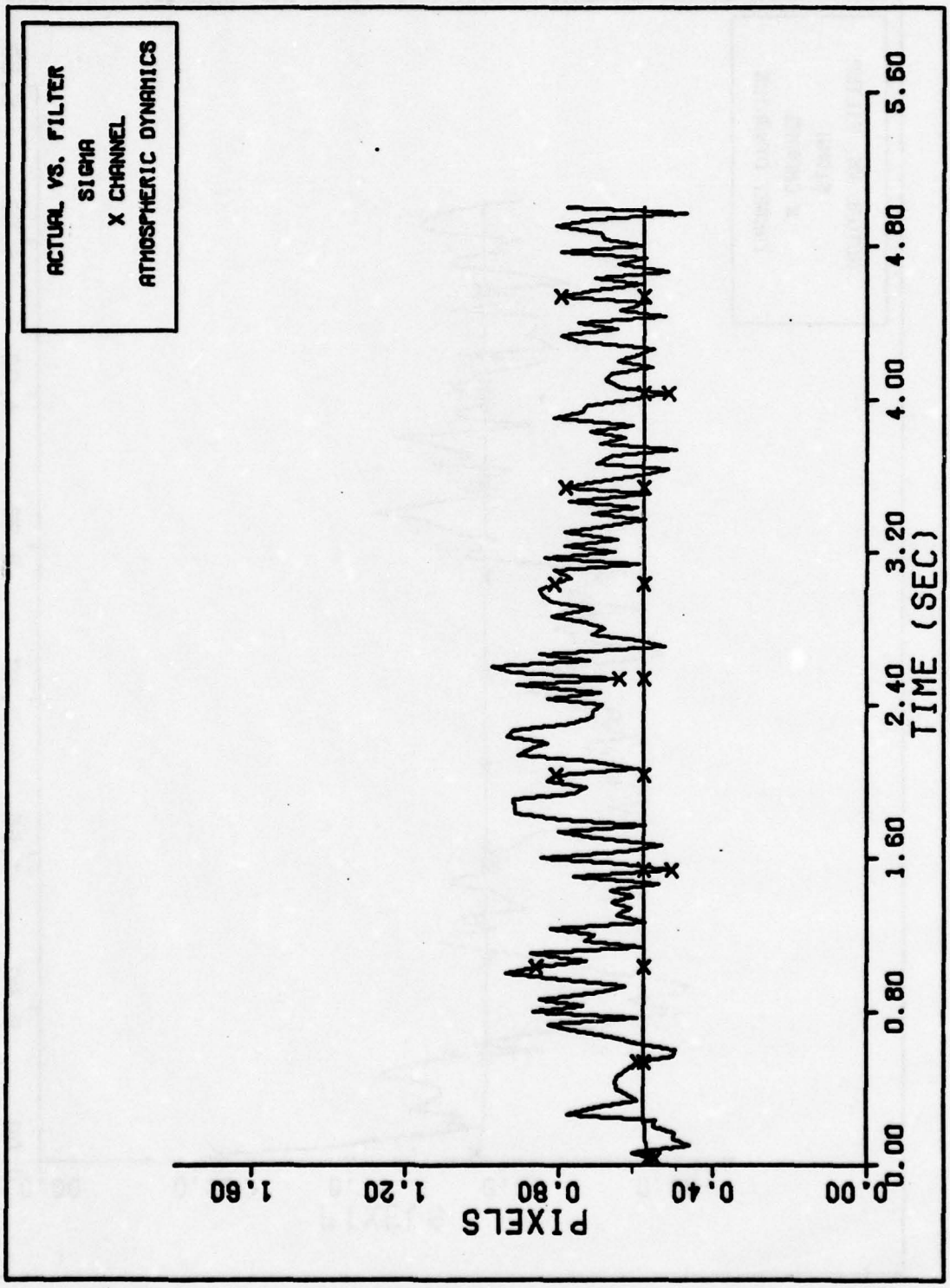
X CHANNEL ATMOSPHERICS ERROR (S/N= 10)

Figure 13a. Case 13 Performance Plot



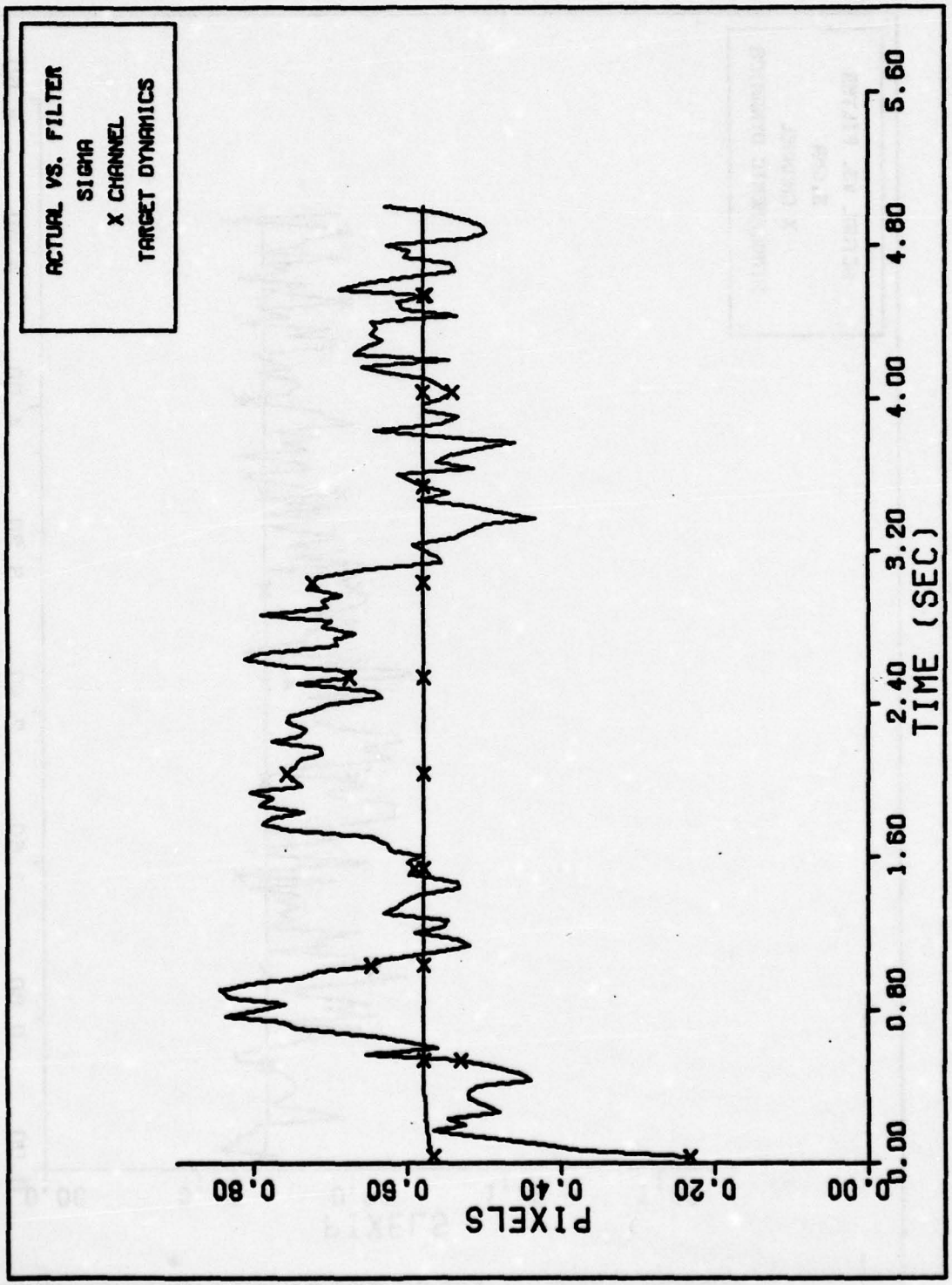
X CHANNEL DYNAMICS ERROR (S/N= 10)

Figure 13b. Case 13 Performance Plot



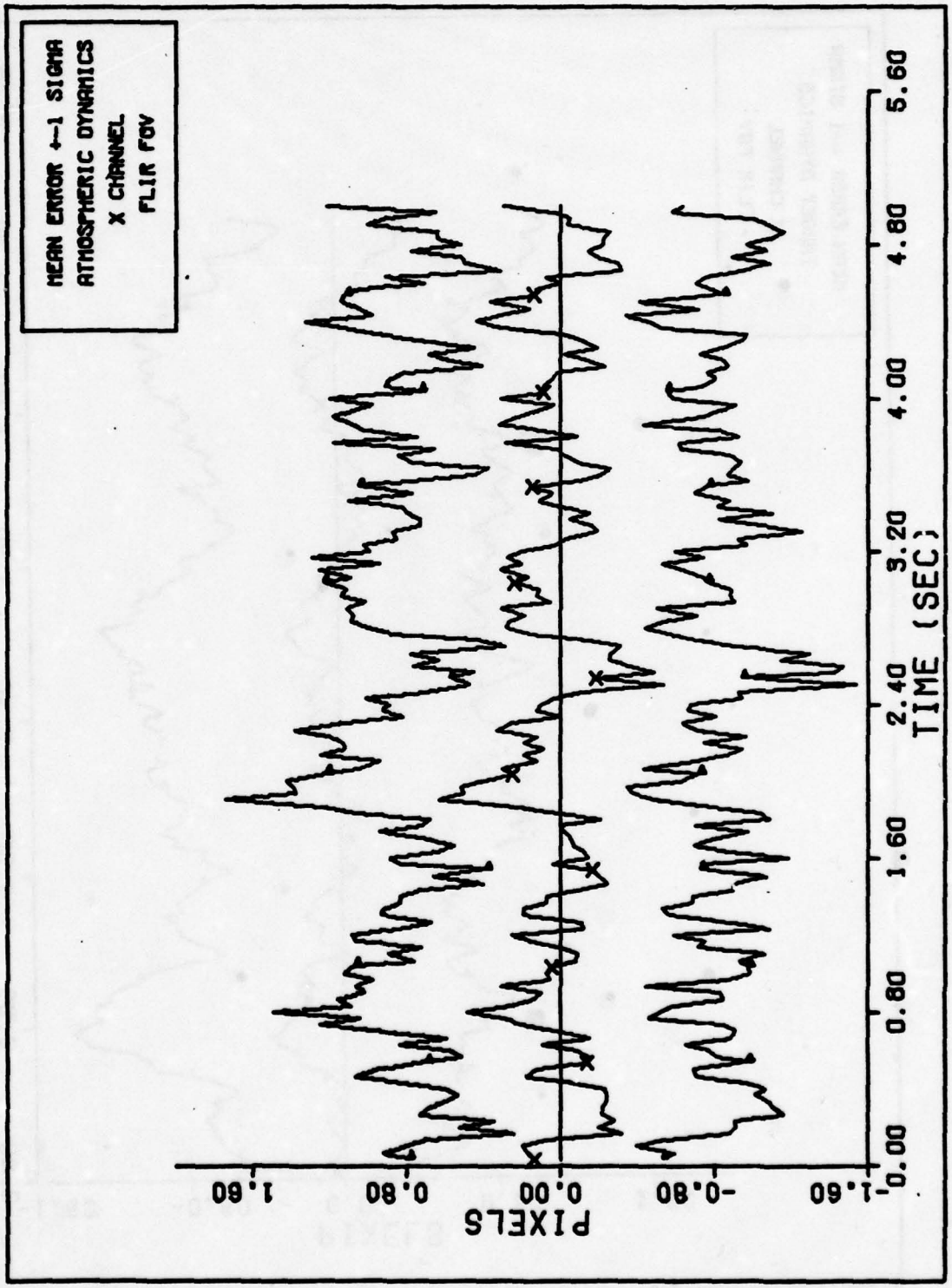
FILTER VS. ACTUAL SIGMA PLOT (S/N = 10)

Figure 13c. Case 13 Performance Plot



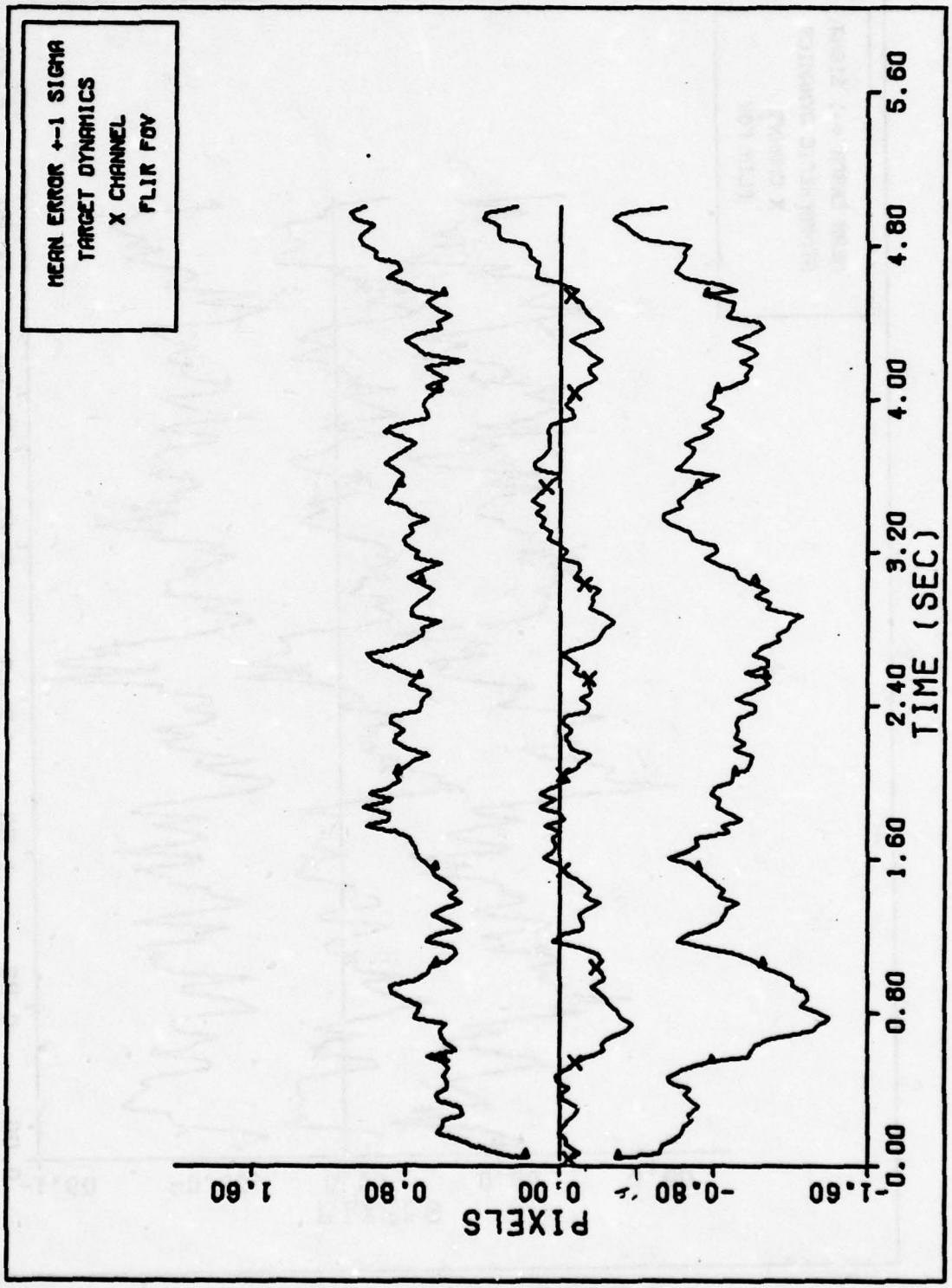
FILTER VS. ACTUAL SIGMA PLOT (S/N = 10)

Figure 13d. Case 13 Performance Plot



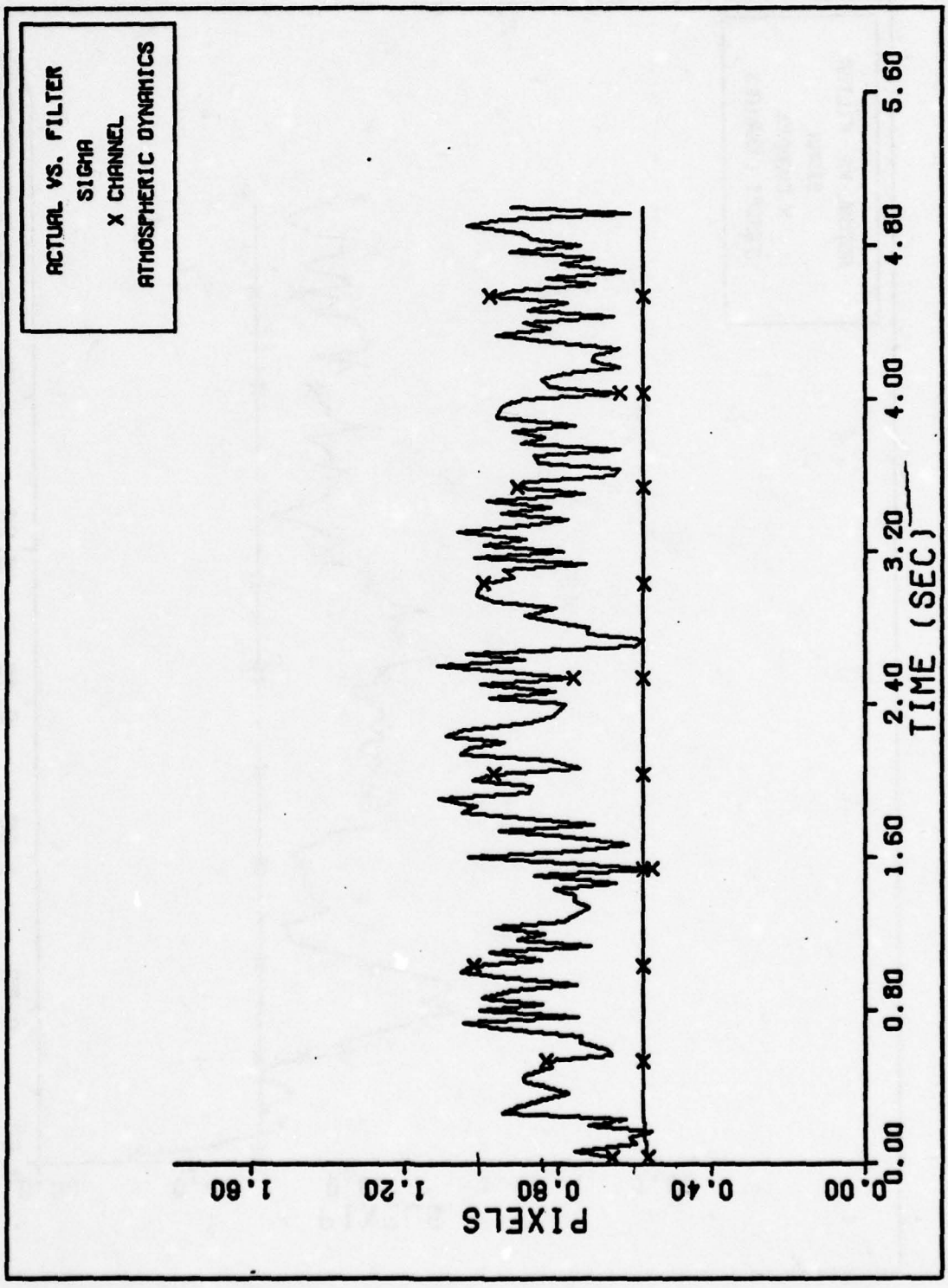
X CHANNEL ATMOSPHERICS ERROR (S/N= 10)

Figure 14a. Case 14 Performance Plot



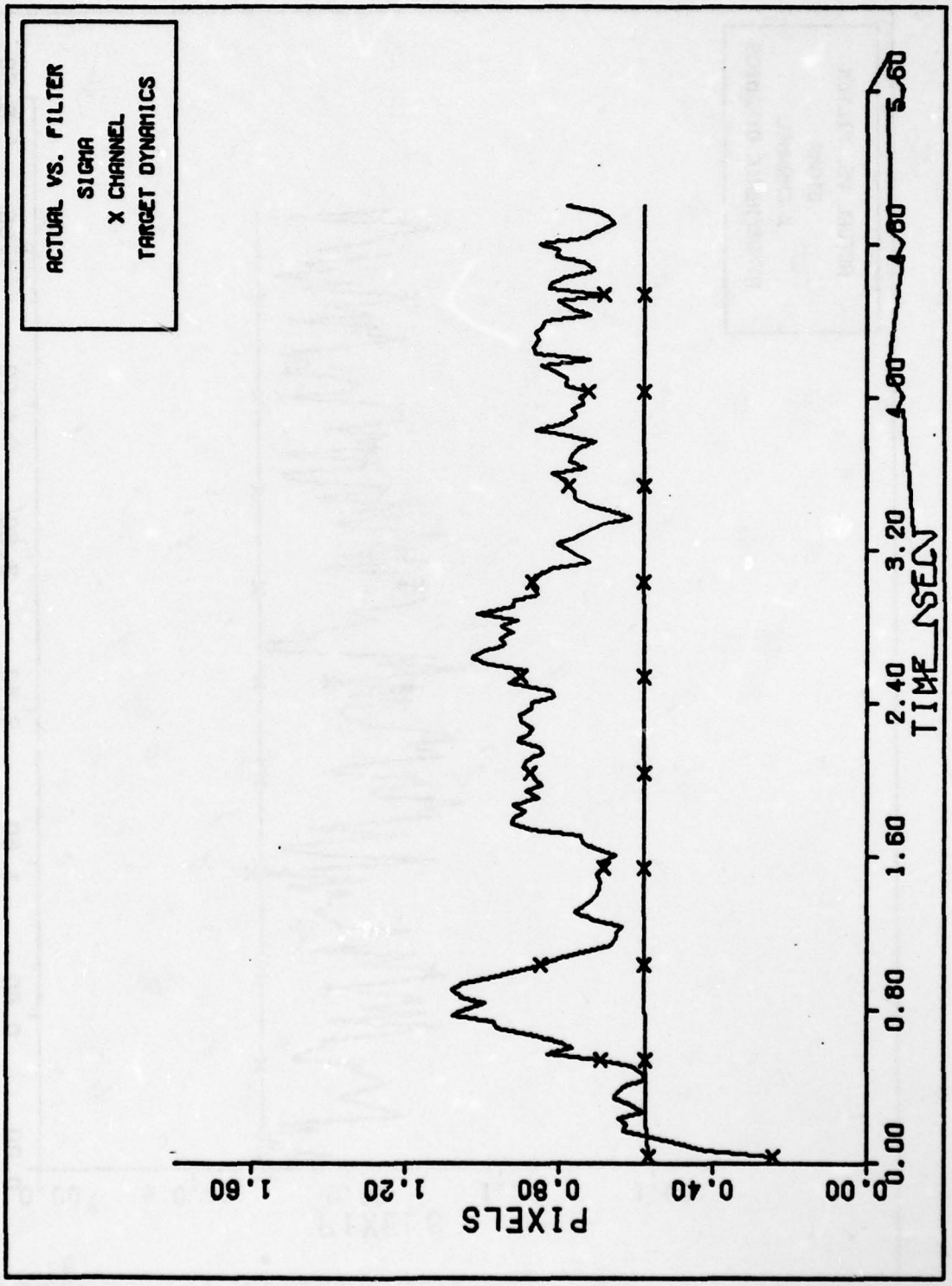
X CHANNEL DYNAMICS ERROR (S/N= 10)

Figure 14b. Case 14 Performance Plot



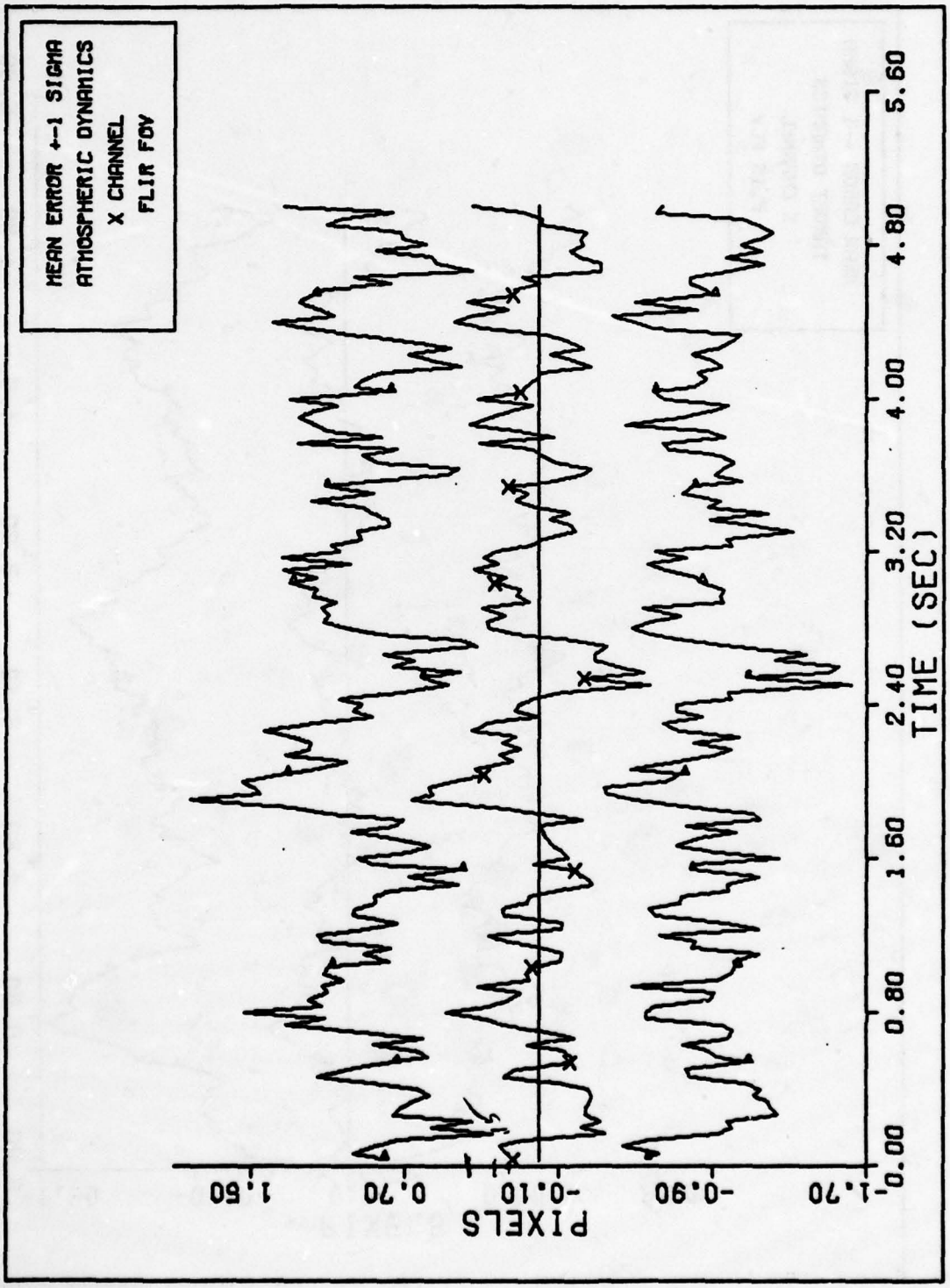
FILTER VS. ACTUAL SIGMA PLOT (S/N = 10)

Figure 14c. Case 14 Performance Plot



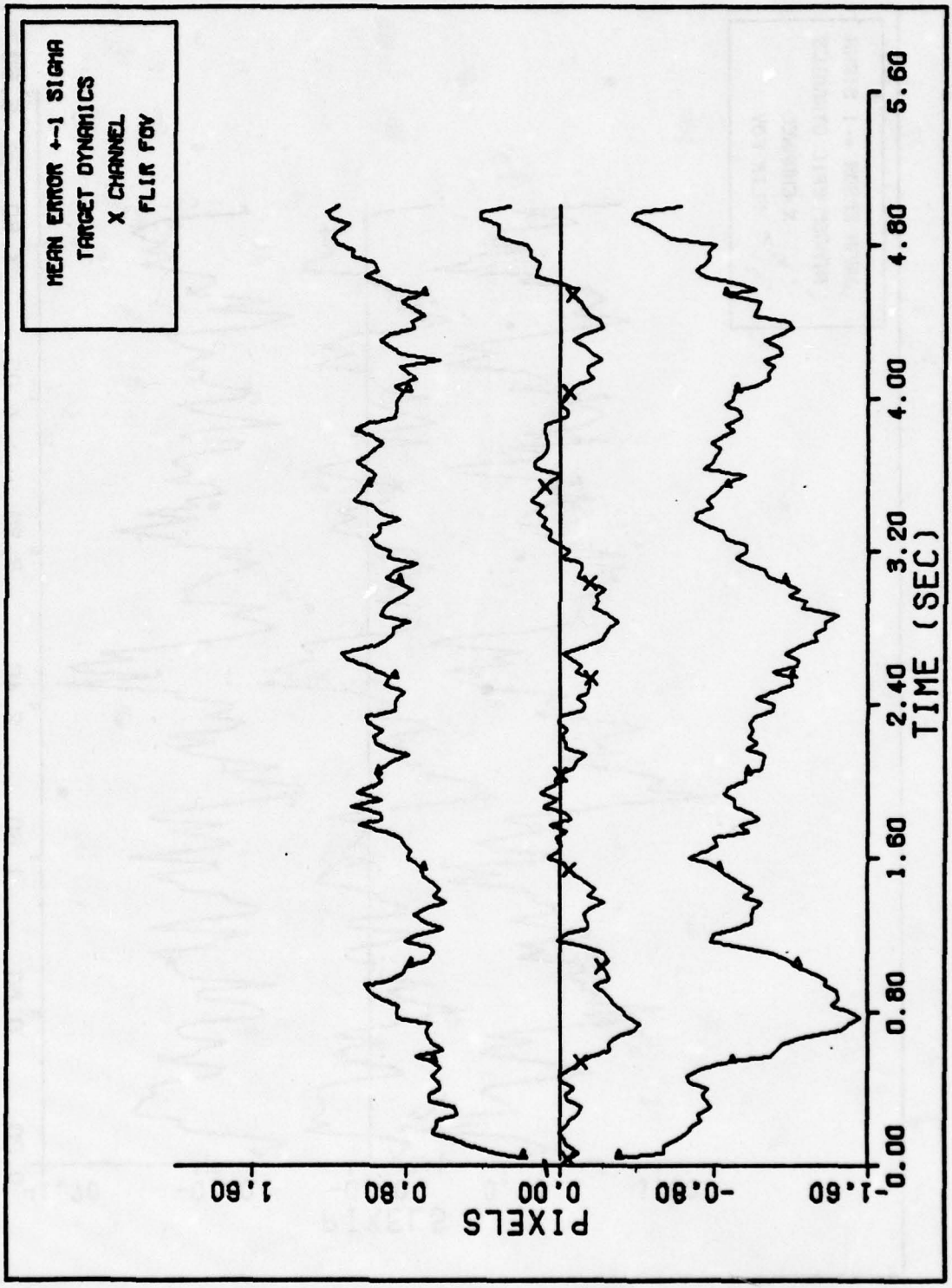
FILTER VS. ACTUAL SIGMA PLOT (S/N = 10)

Figure 14d. Case 14 Performance Plot



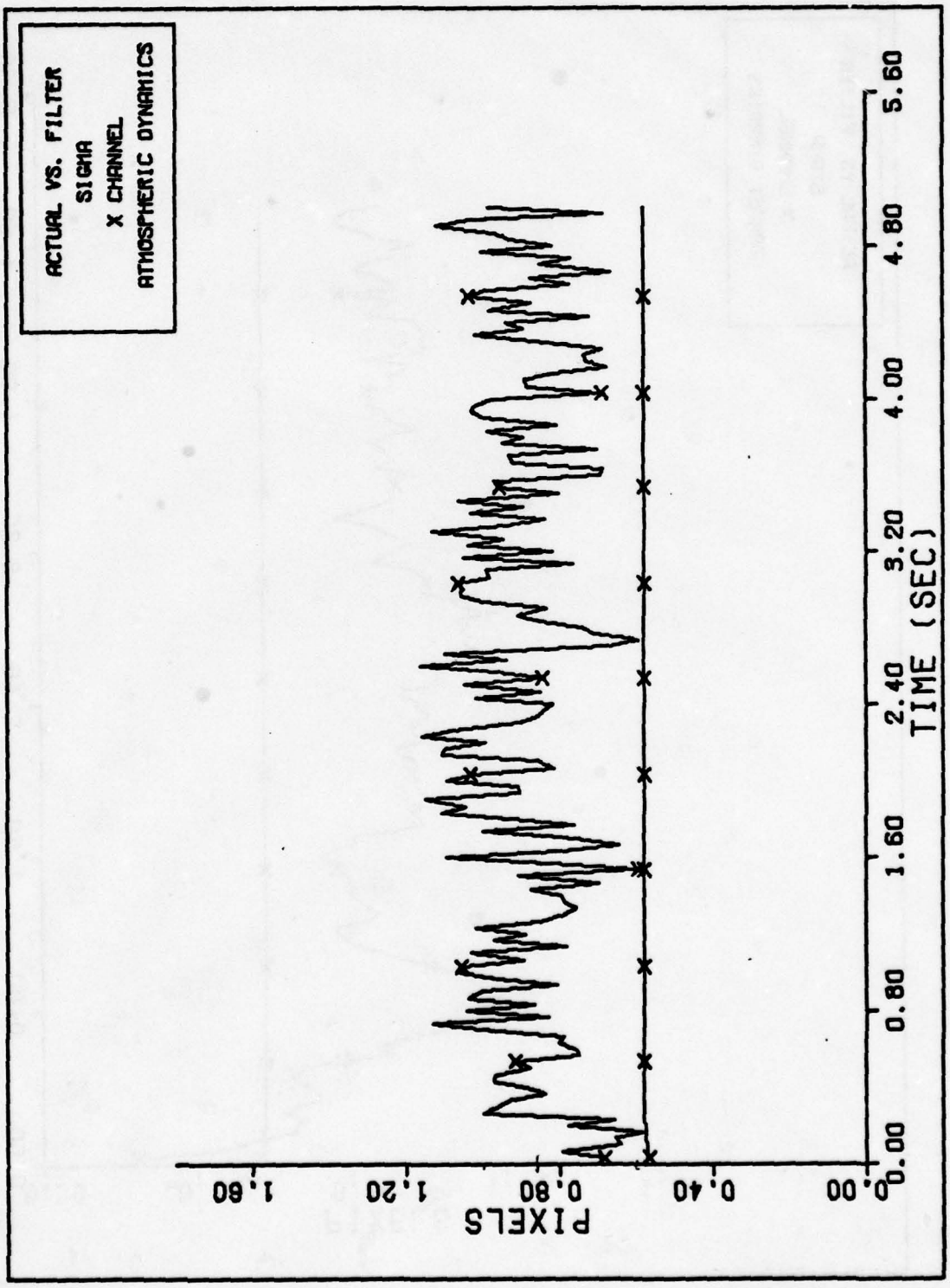
X CHANNEL ATMOSPHERICS ERROR (S/N= 10)

Figure 15a. Case 15 Performance Plot



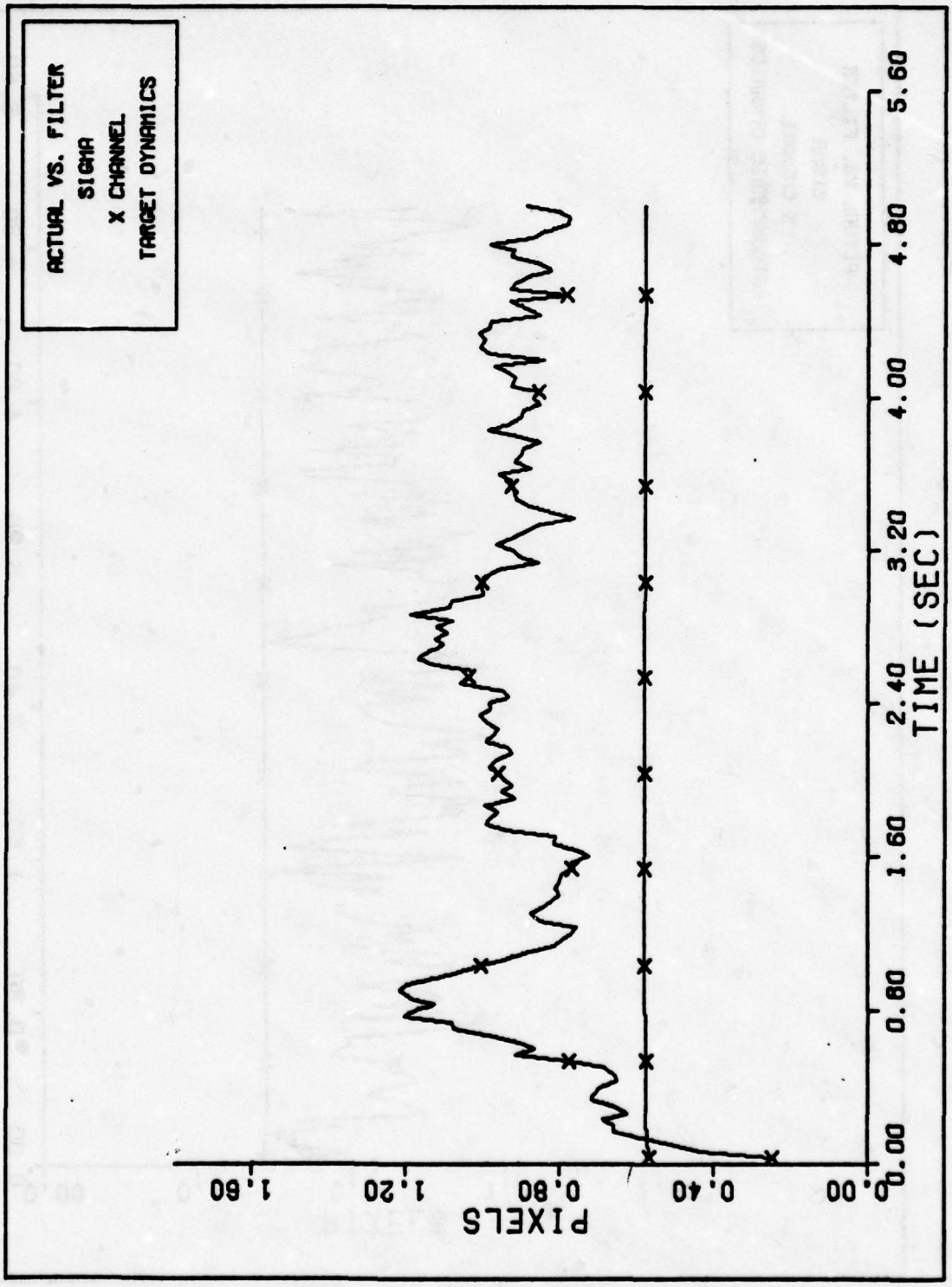
X CHANNEL DYNAMICS ERROR (S/N= 10)

Figure 15b. Case 15 Performance Plot



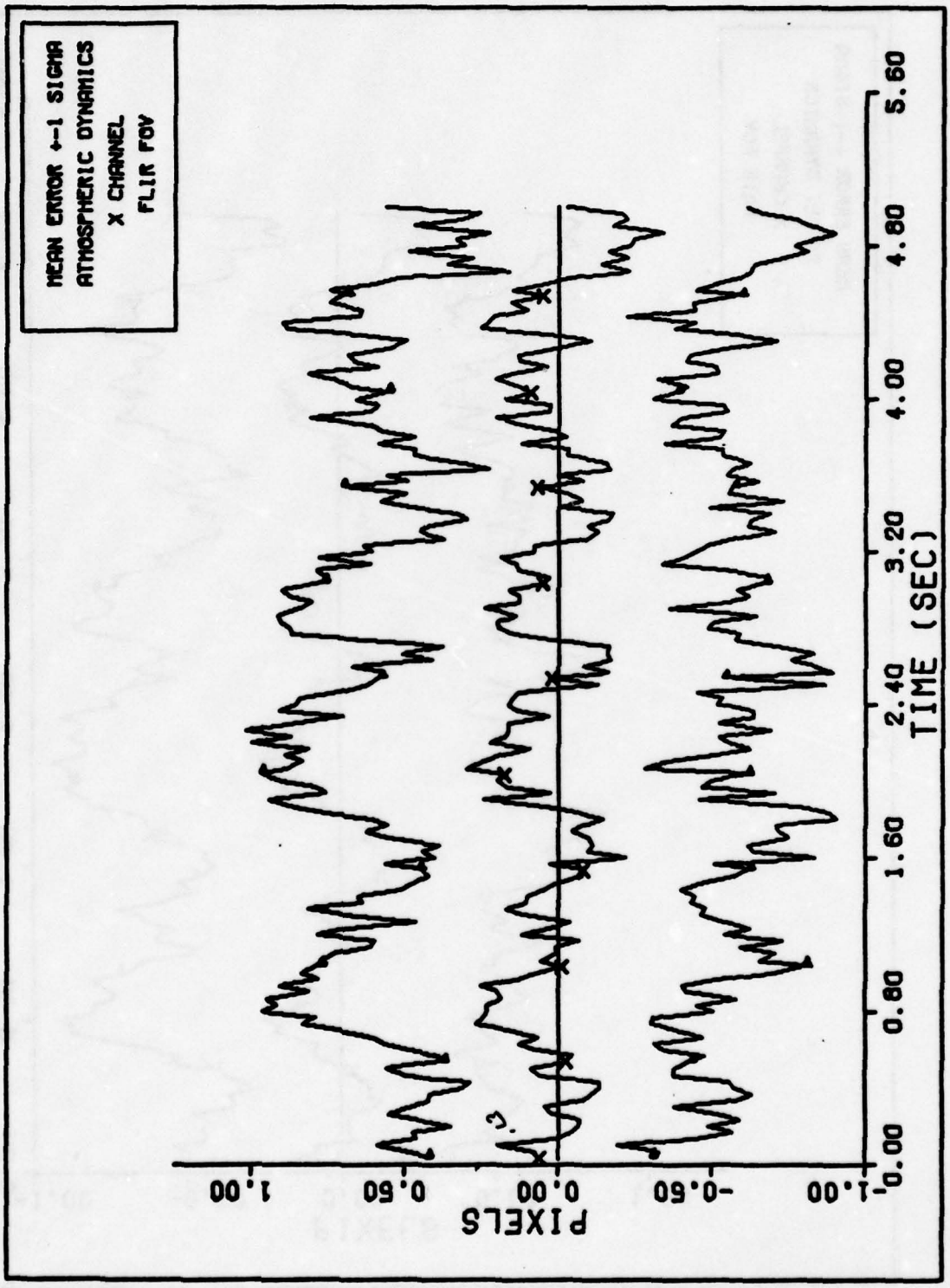
FILTER VS. ACTUAL SIGMA PLOT (S/N = 10)

Figure 15c. Case 15 Performance Plot



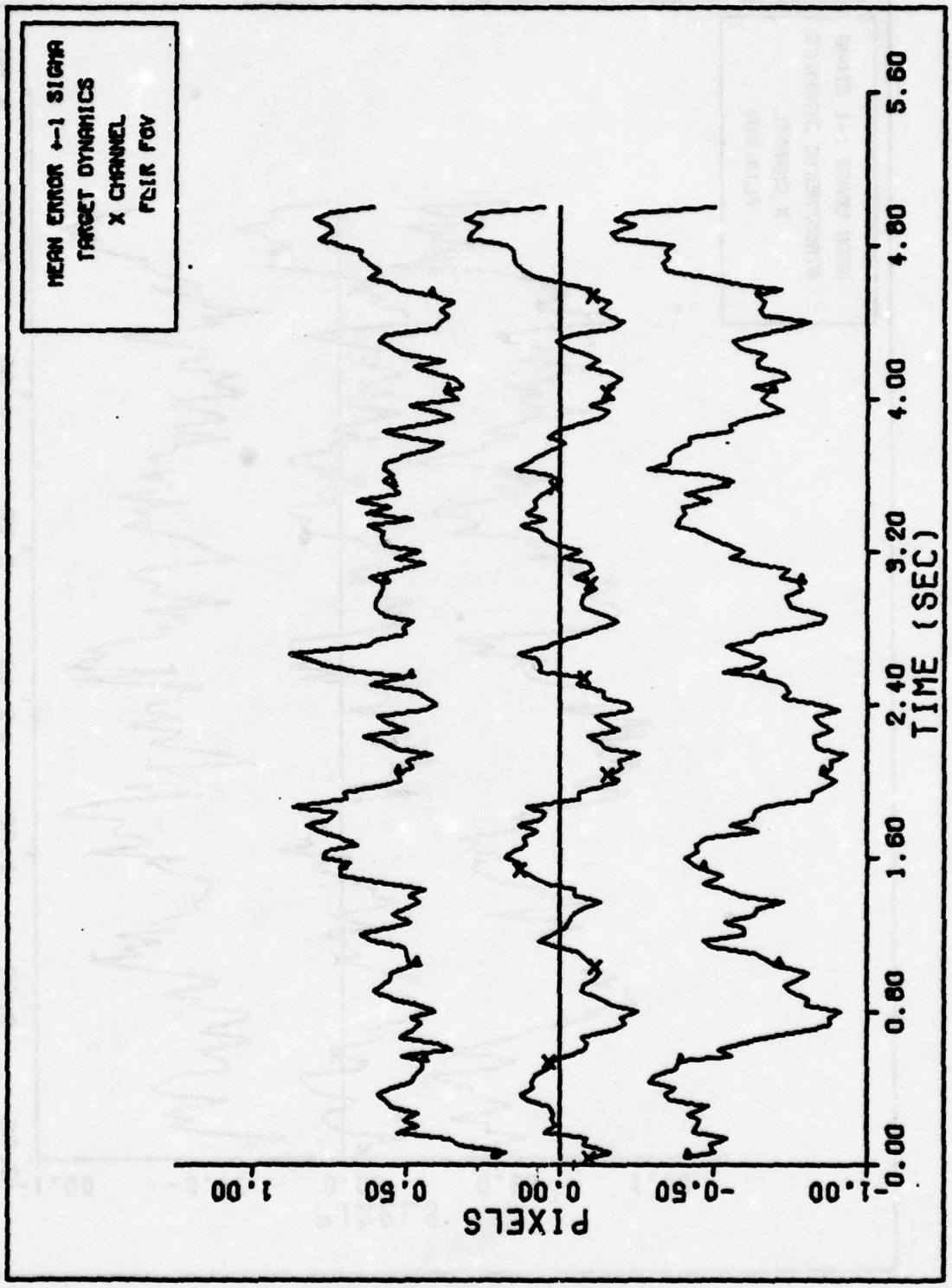
FILTER VS. ACTUAL SIGMA PLOT (S/N = 10)

Figure 15d. Case 15 Performance Plot



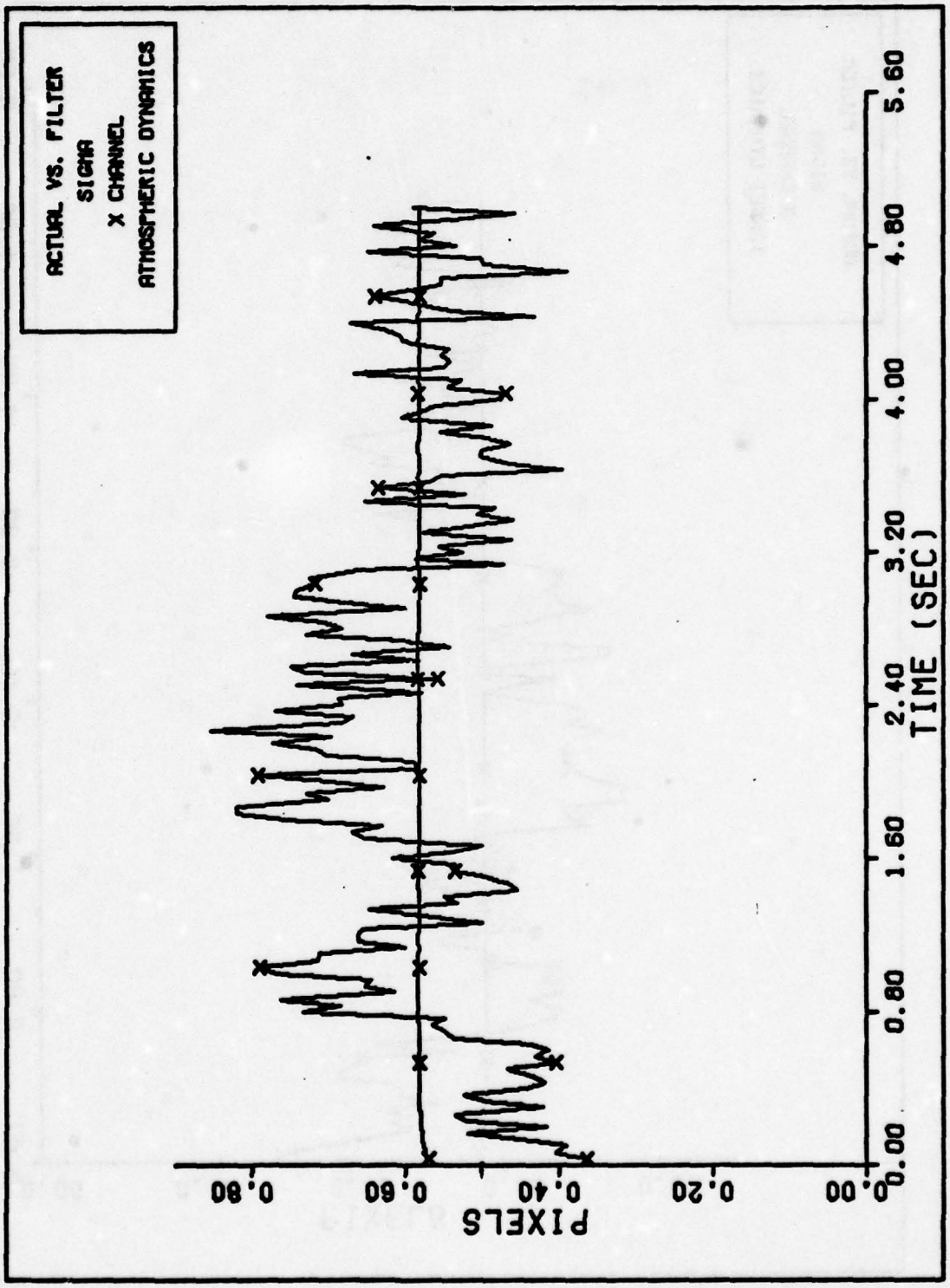
X CHANNEL ATMOSPHERICS ERROR (S/N= 10)

Figure 16a. Case 16 Performance Plot



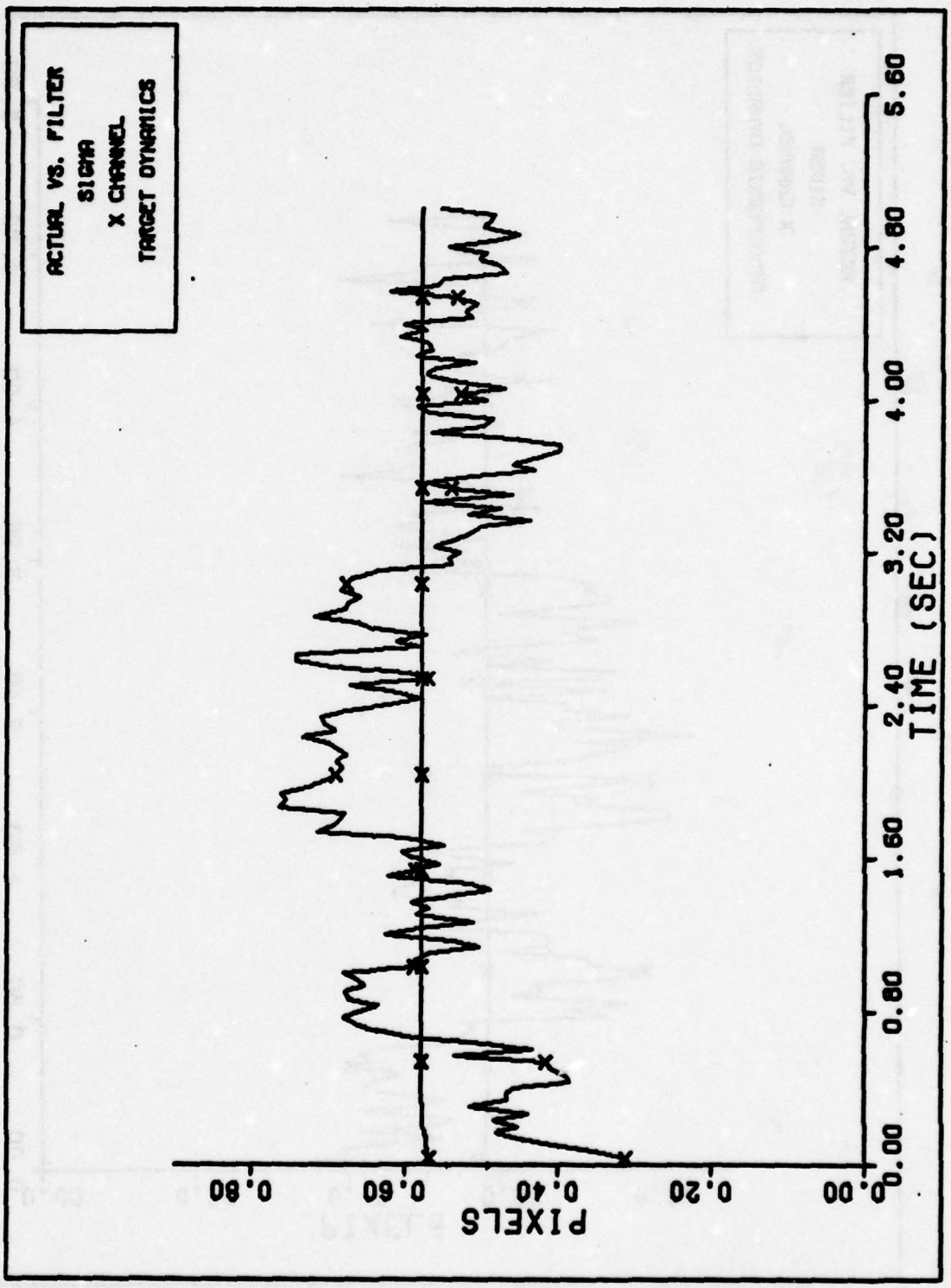
X CHANNEL DYNAMICS ERROR (S/N= 10)

Figure 16b. Case 16 Performance Plot



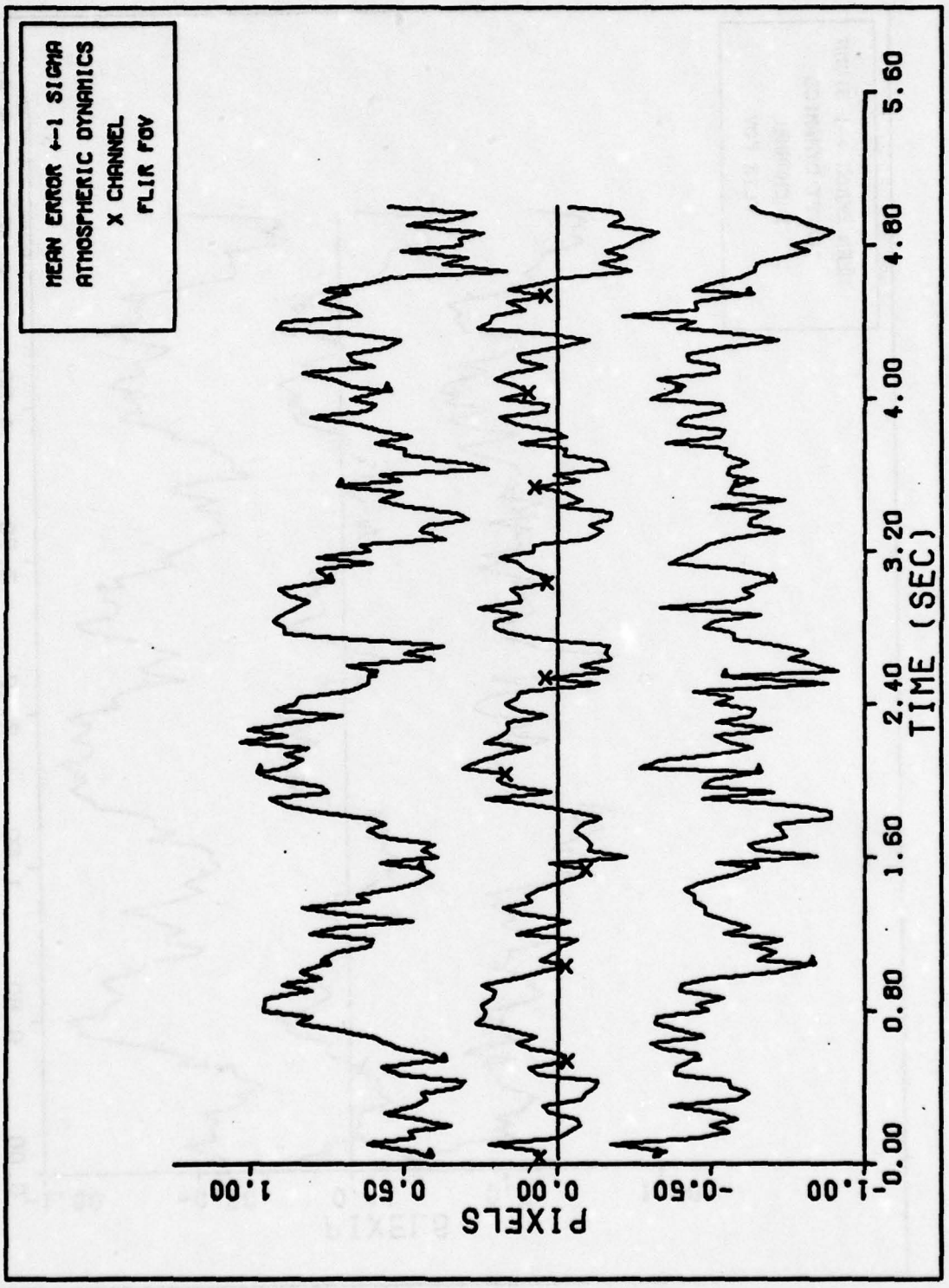
FILTER VS. ACTUAL SIGMA PLOT (S/N = 10)

Figure 16c. Case 16 Performance Plot



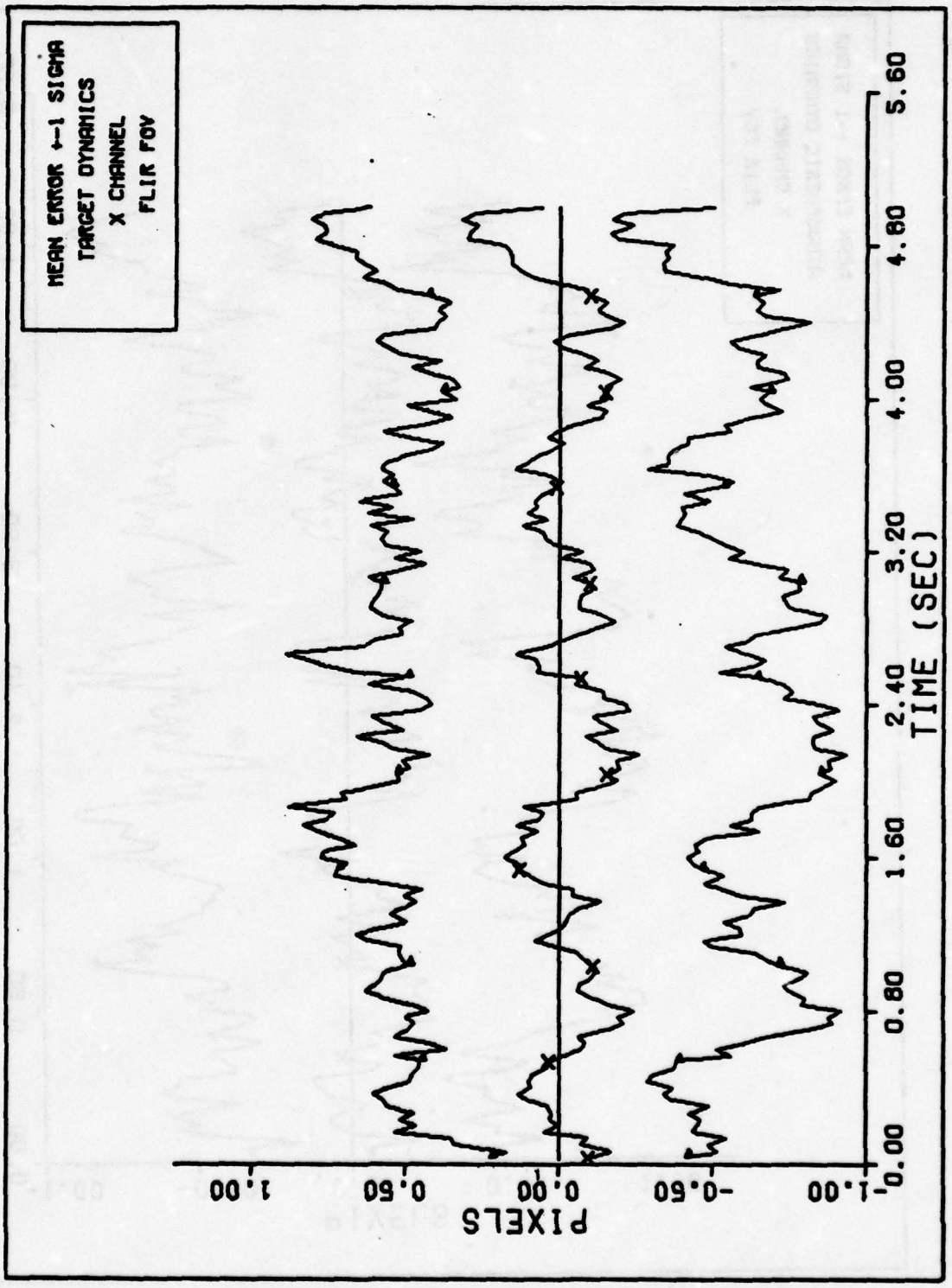
FILTER VS. ACTUAL SIGMA PLOT (S/N = 10)

Figure 16d. Case 16 Performance Plot



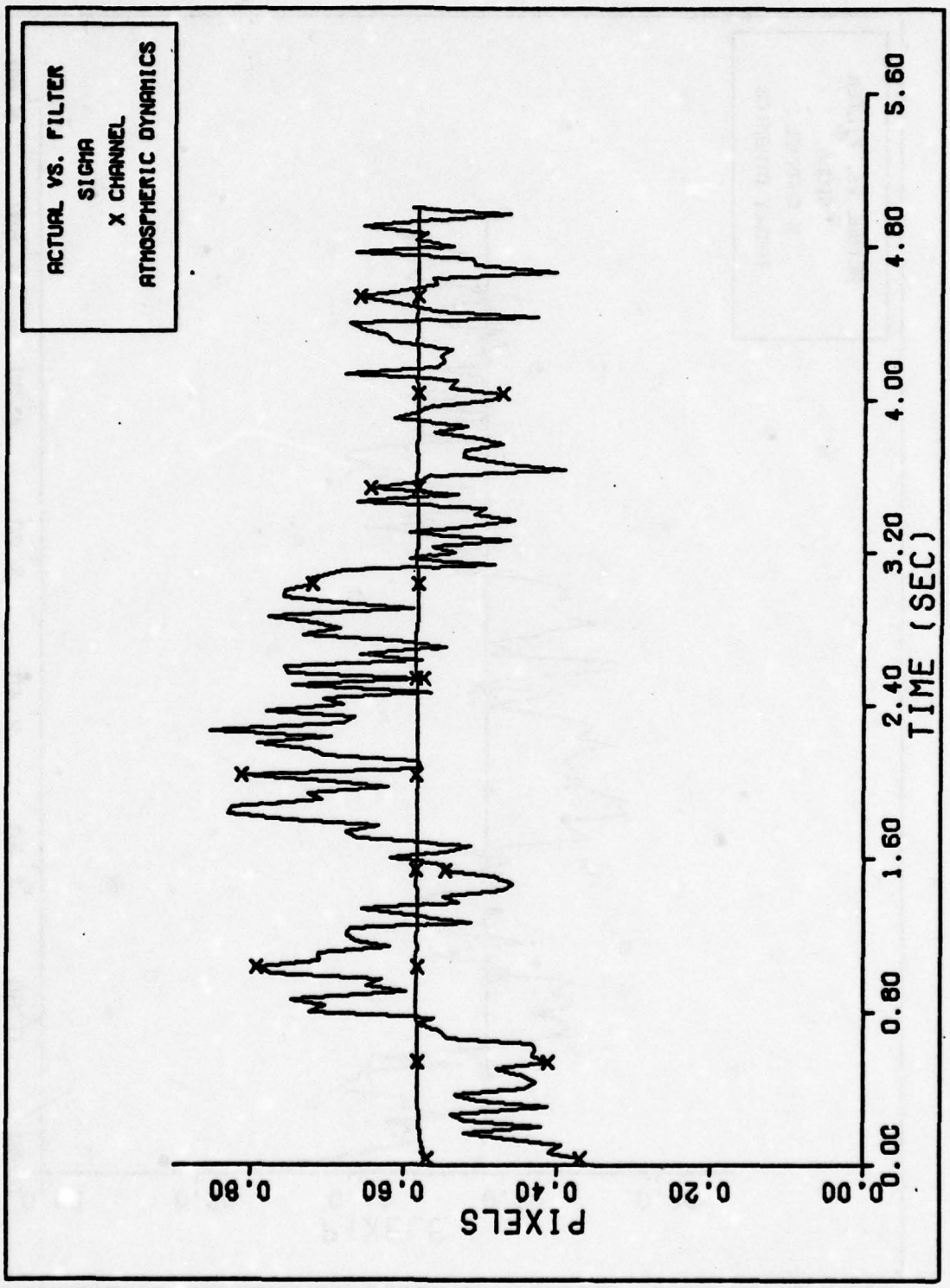
X CHANNEL ATMOSPHERICS ERROR (S/N= 10)

Figure 17a. Case 17 Performance Plot



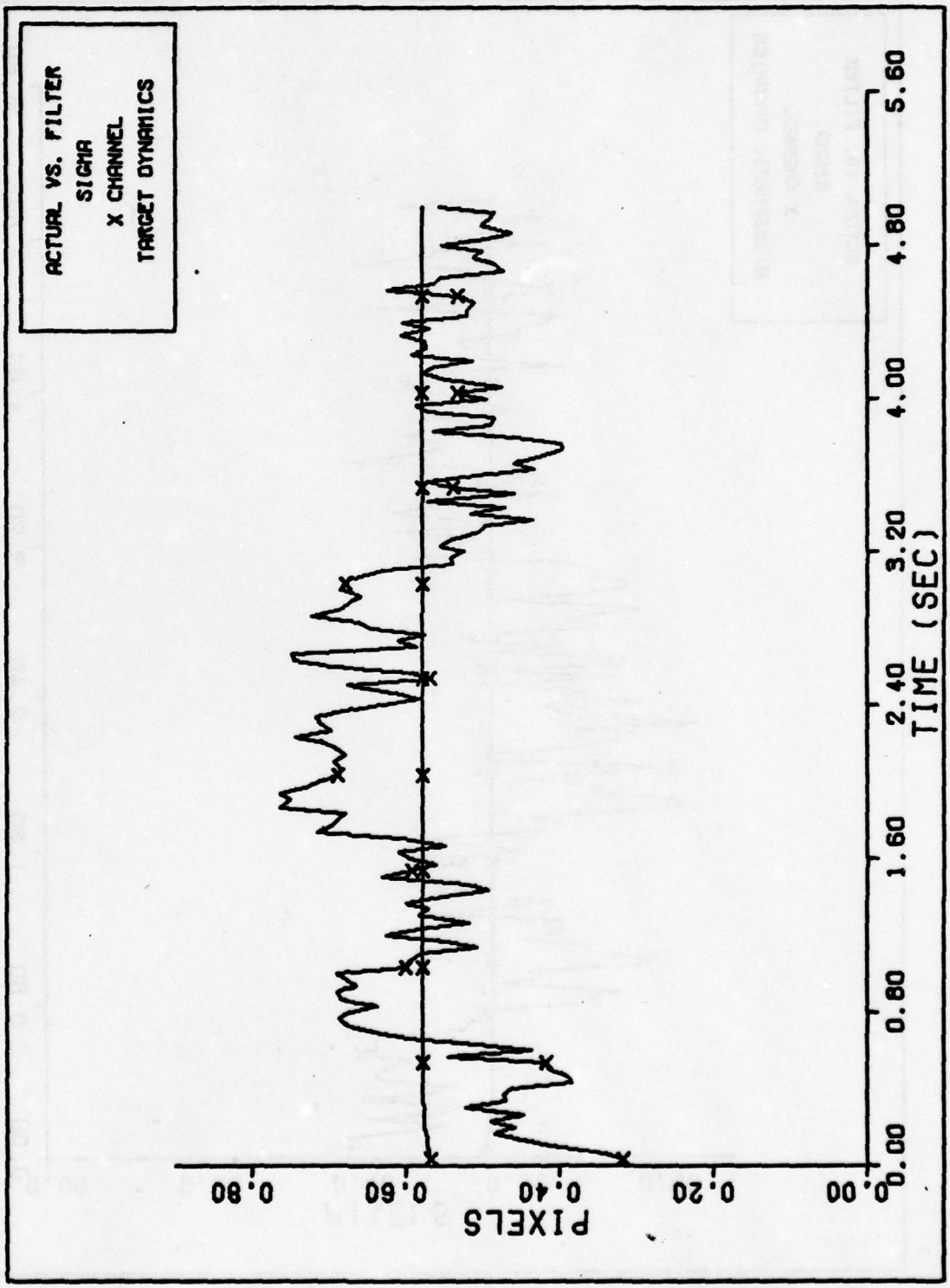
X CHANNEL DYNAMICS ERROR (S/N= 10)

Figure 17b. Case 17 Performance Plot



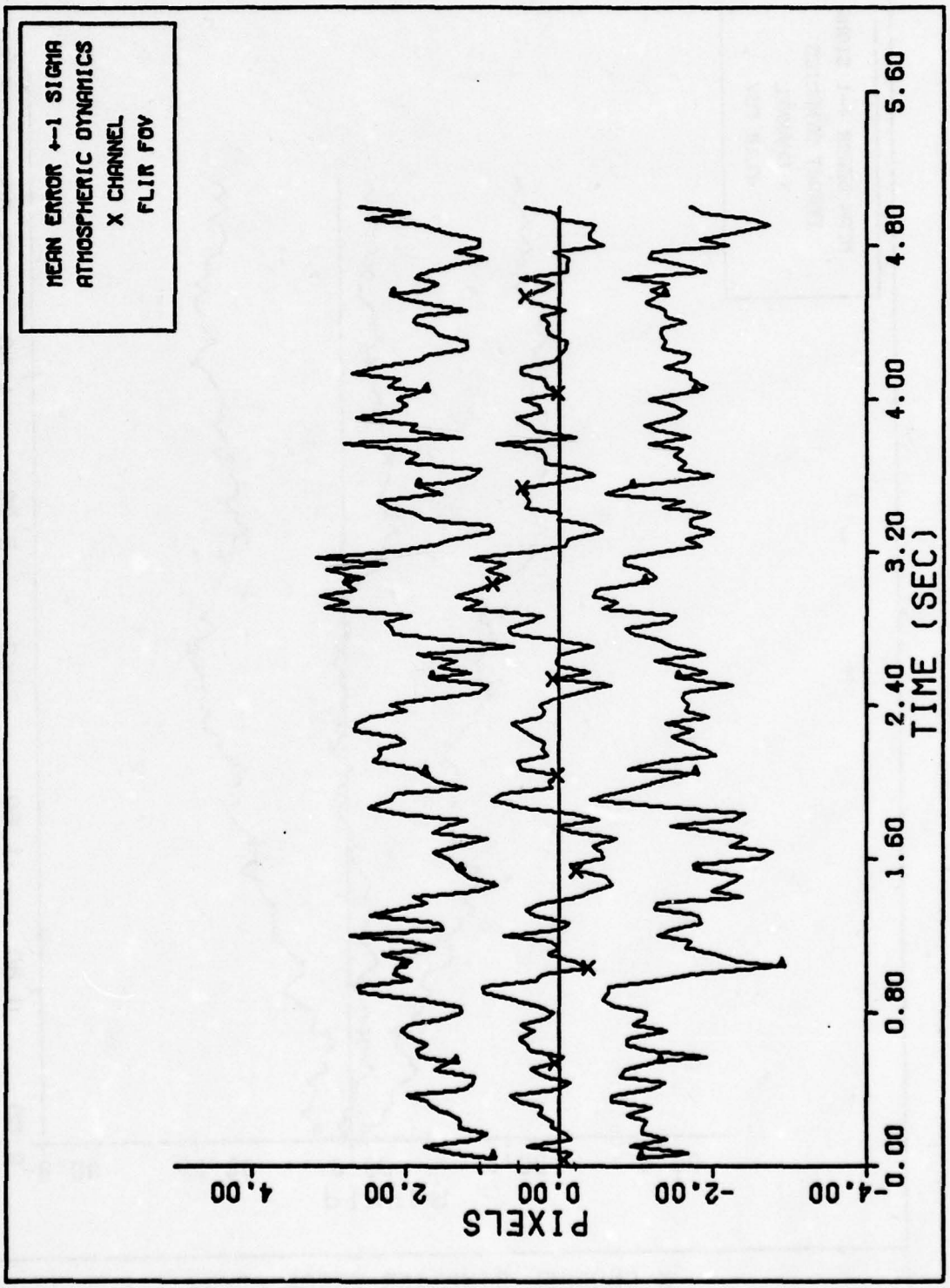
FILTER VS. ACTUAL SIGMA PLOT (S/N = 10)

Figure 17c. Case 17 Performance Plot



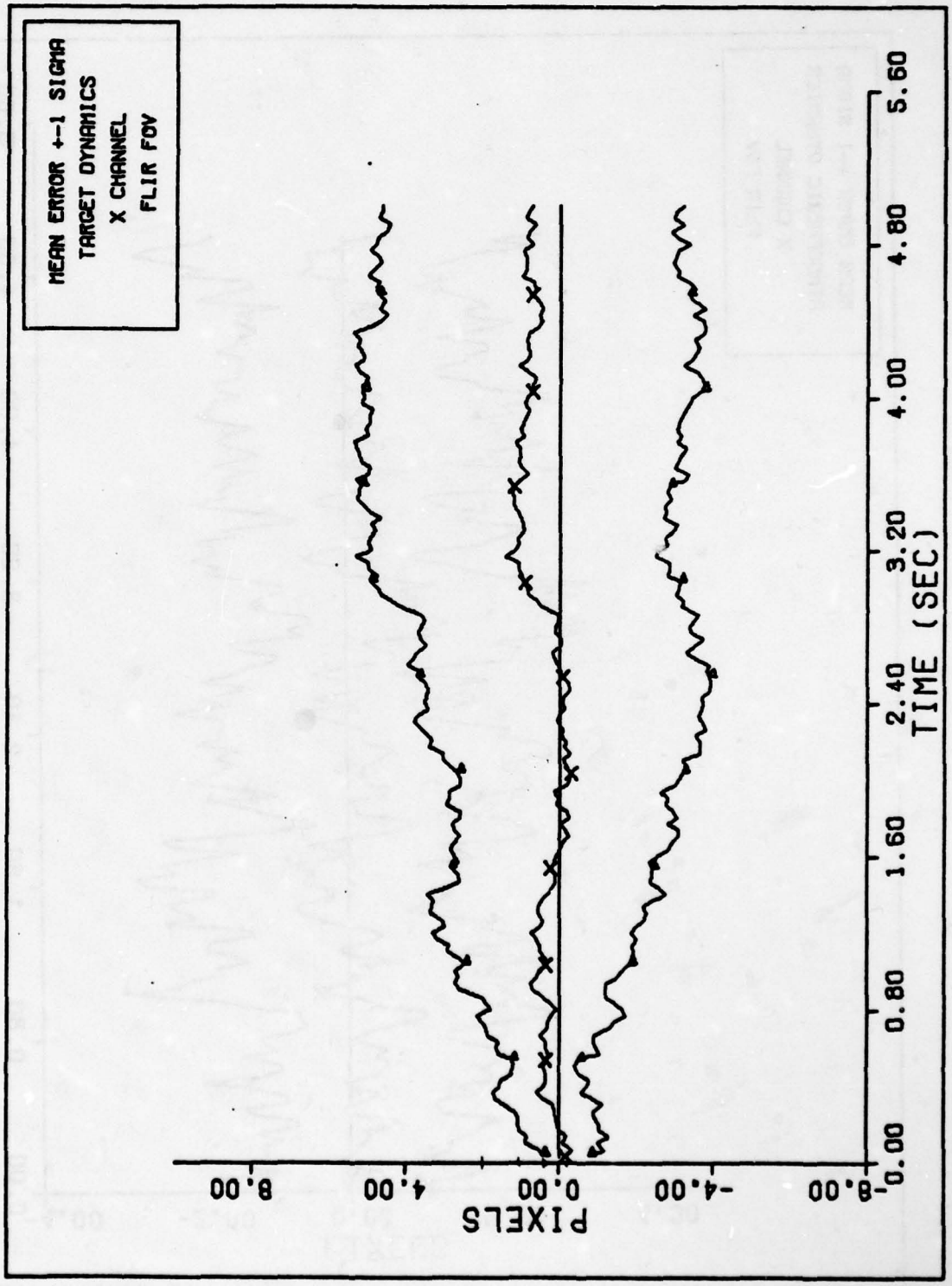
FILTER VS. ACTUAL SIGMA PLOT (S/N = 10)

Figure 17d. Case 17 Performance Plot



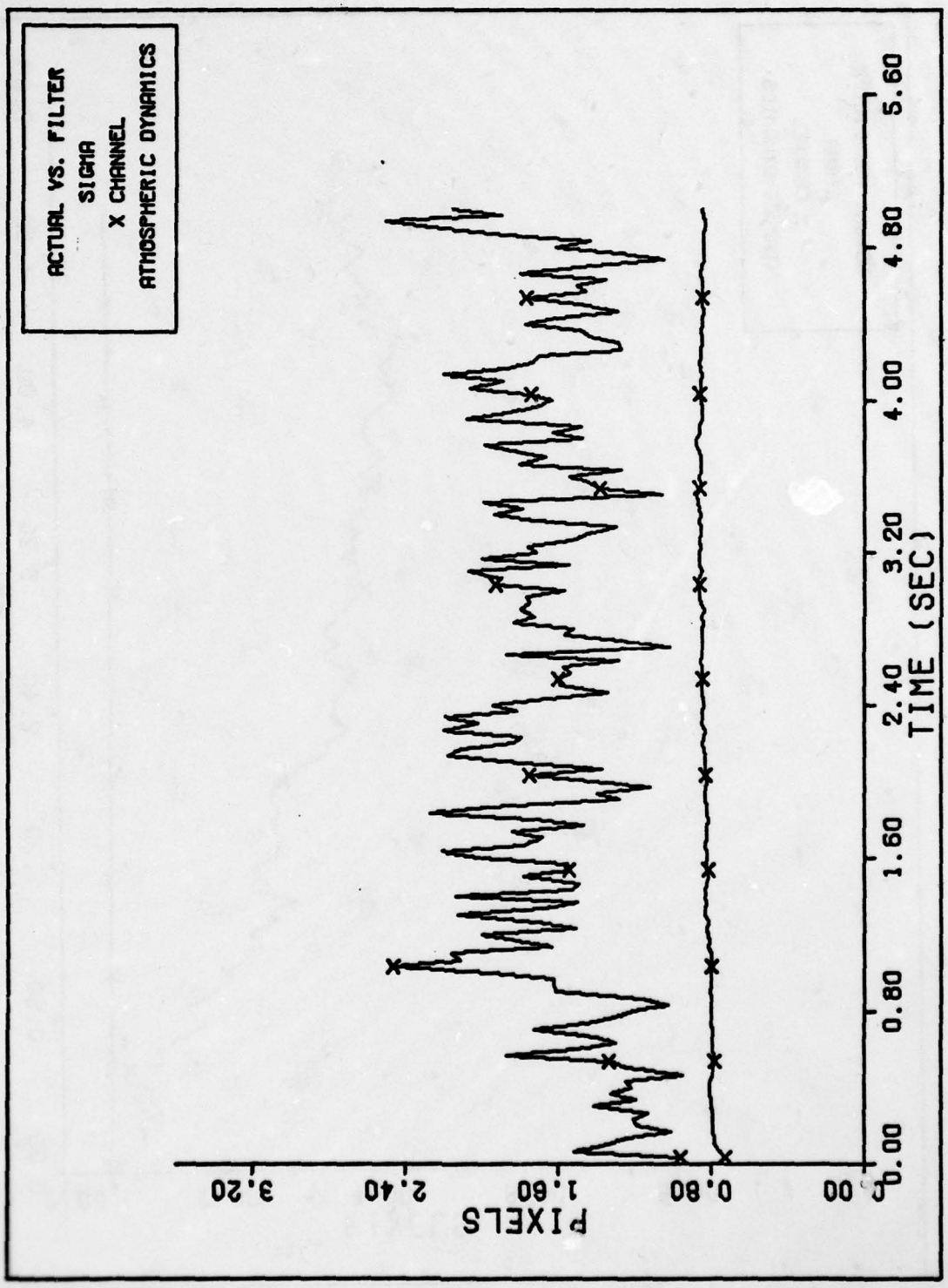
X CHANNEL ATMOSPHERICS ERROR (S/N= 1)

Figure 18a. Case 18 Performance Plot



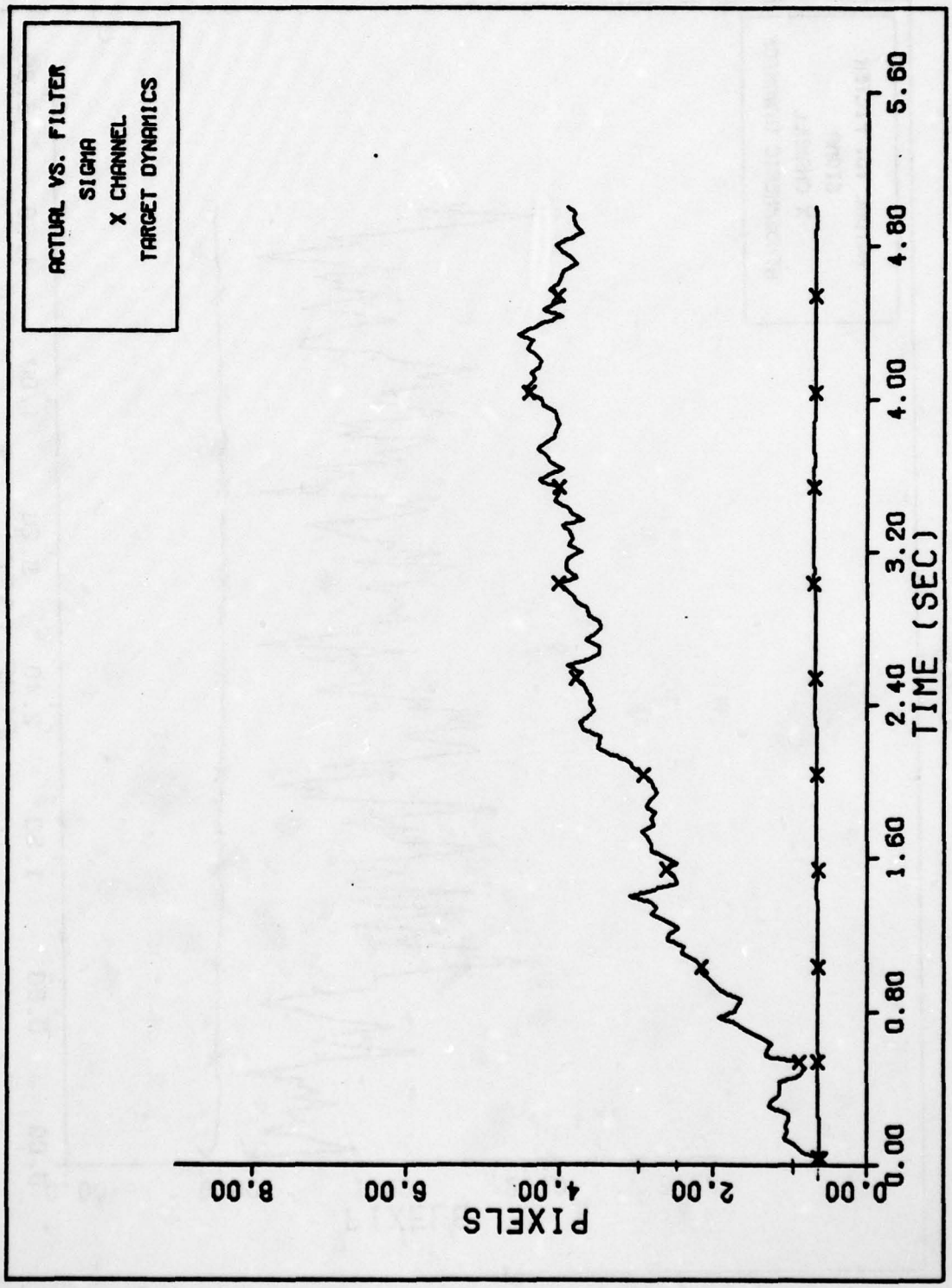
X CHANNEL DYNAMICS ERROR (S/N= 1)

Figure 18b. Case 18 Performance Plot



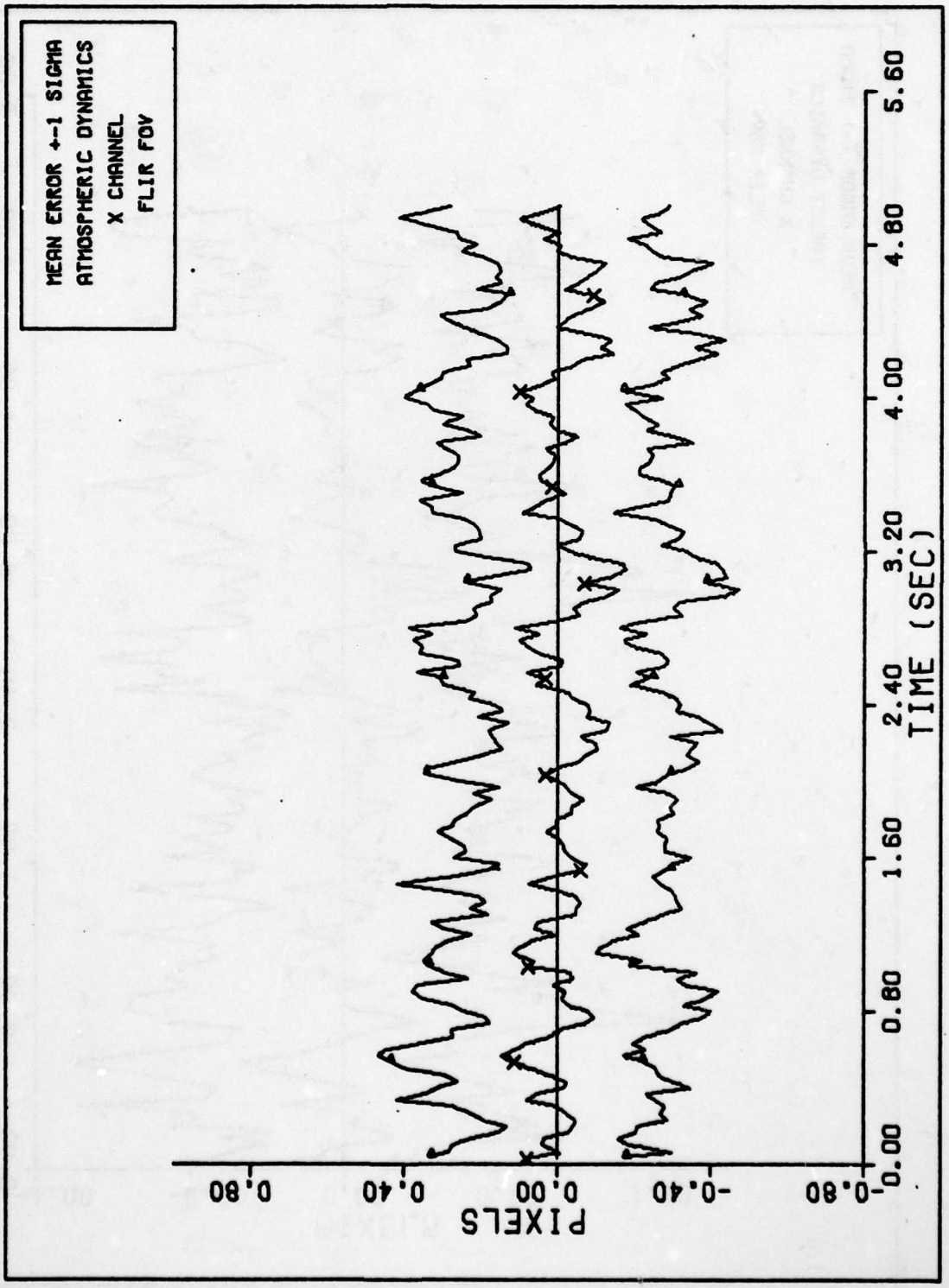
FILTER VS. ACTUAL SIGMA PLOT (S/N = 1)

Figure 18c. Case 18 Performance Plot



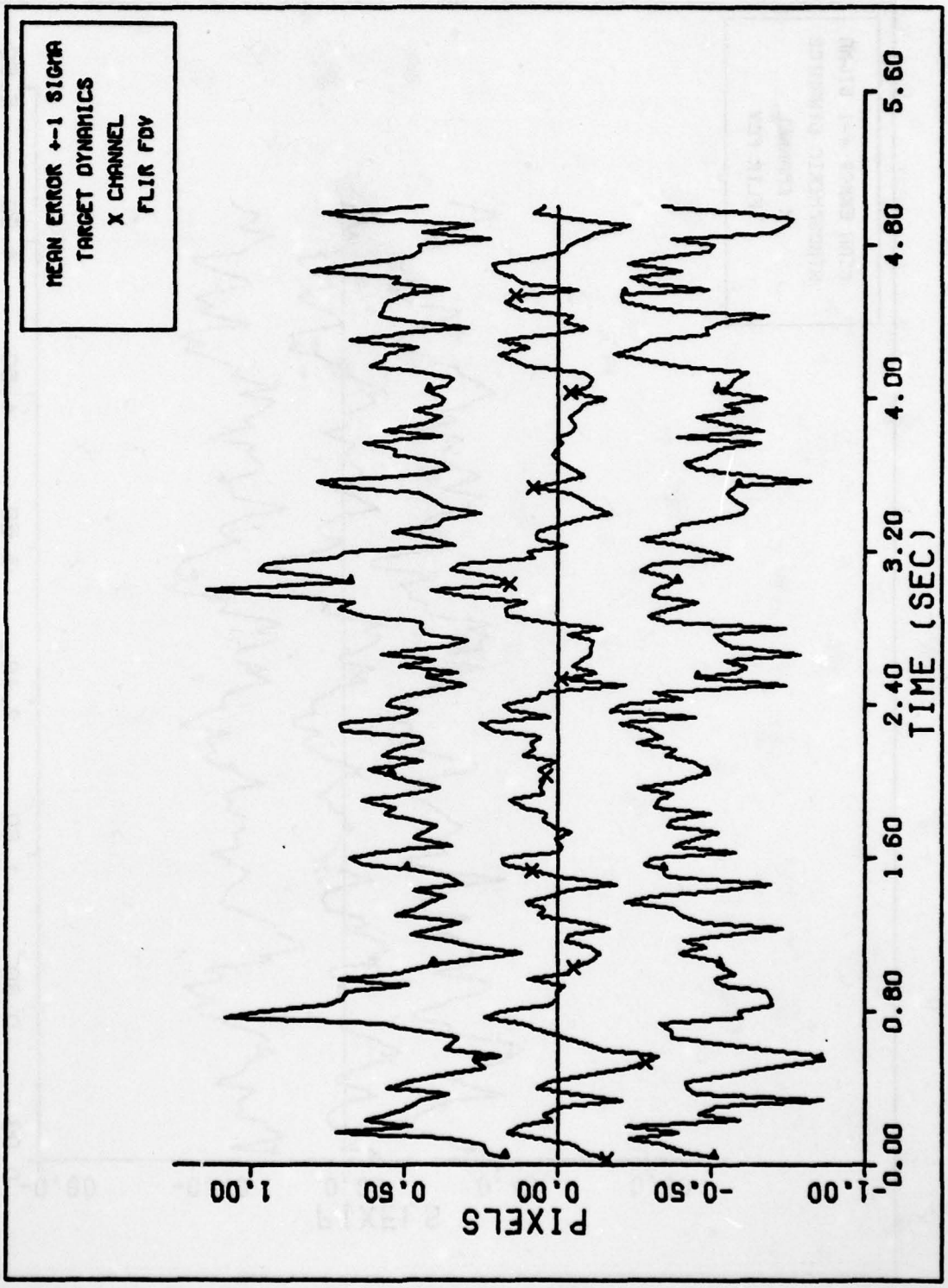
FILTER VS. ACTUAL SIGMA PLOT (S/N = 1)

Figure 18d. Case 18 Performance Plot



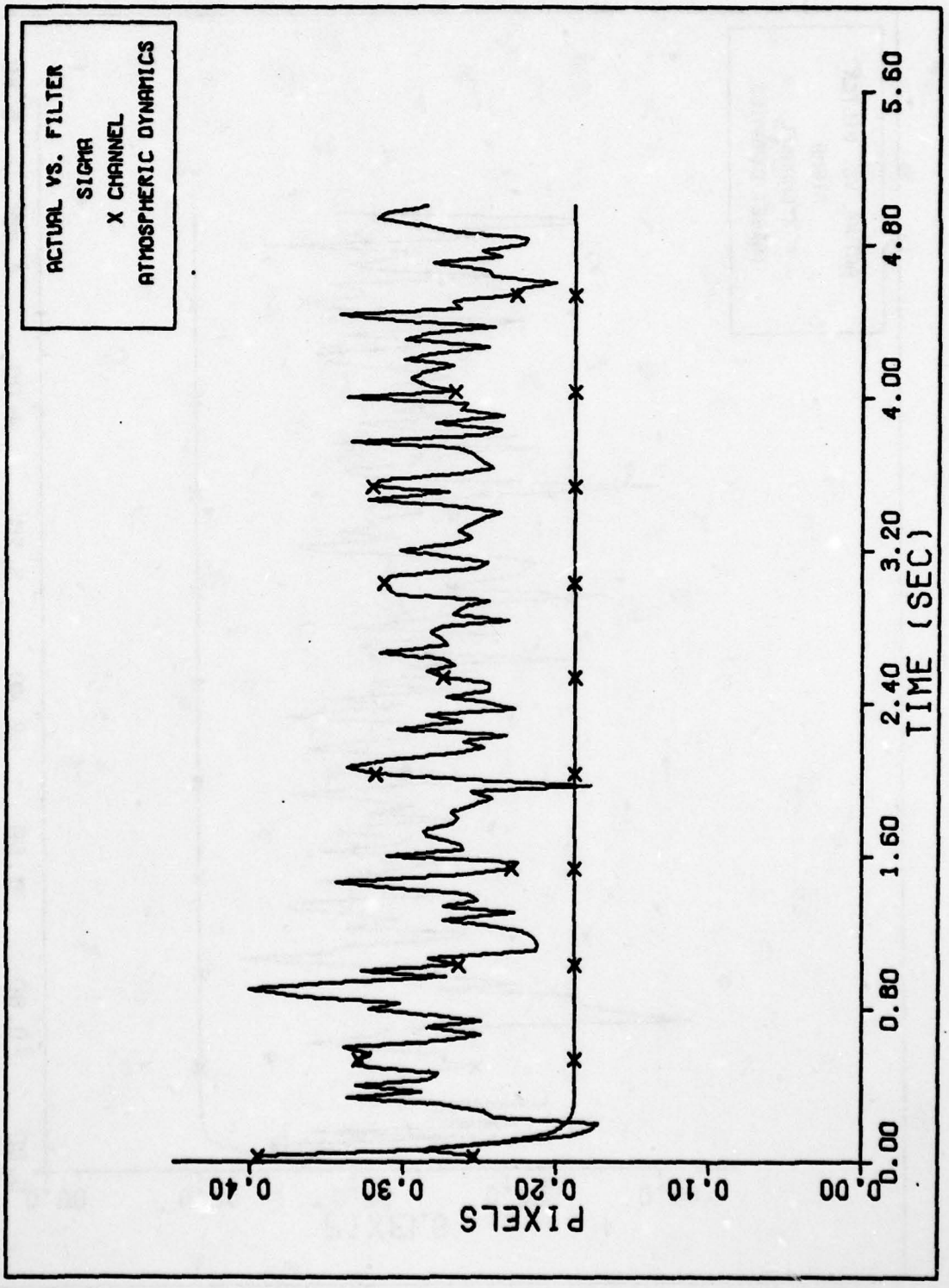
X CHANNEL ATMOSPHERICS ERROR (S/N= 10)

Figure 19a. Case 19 Performance Plot



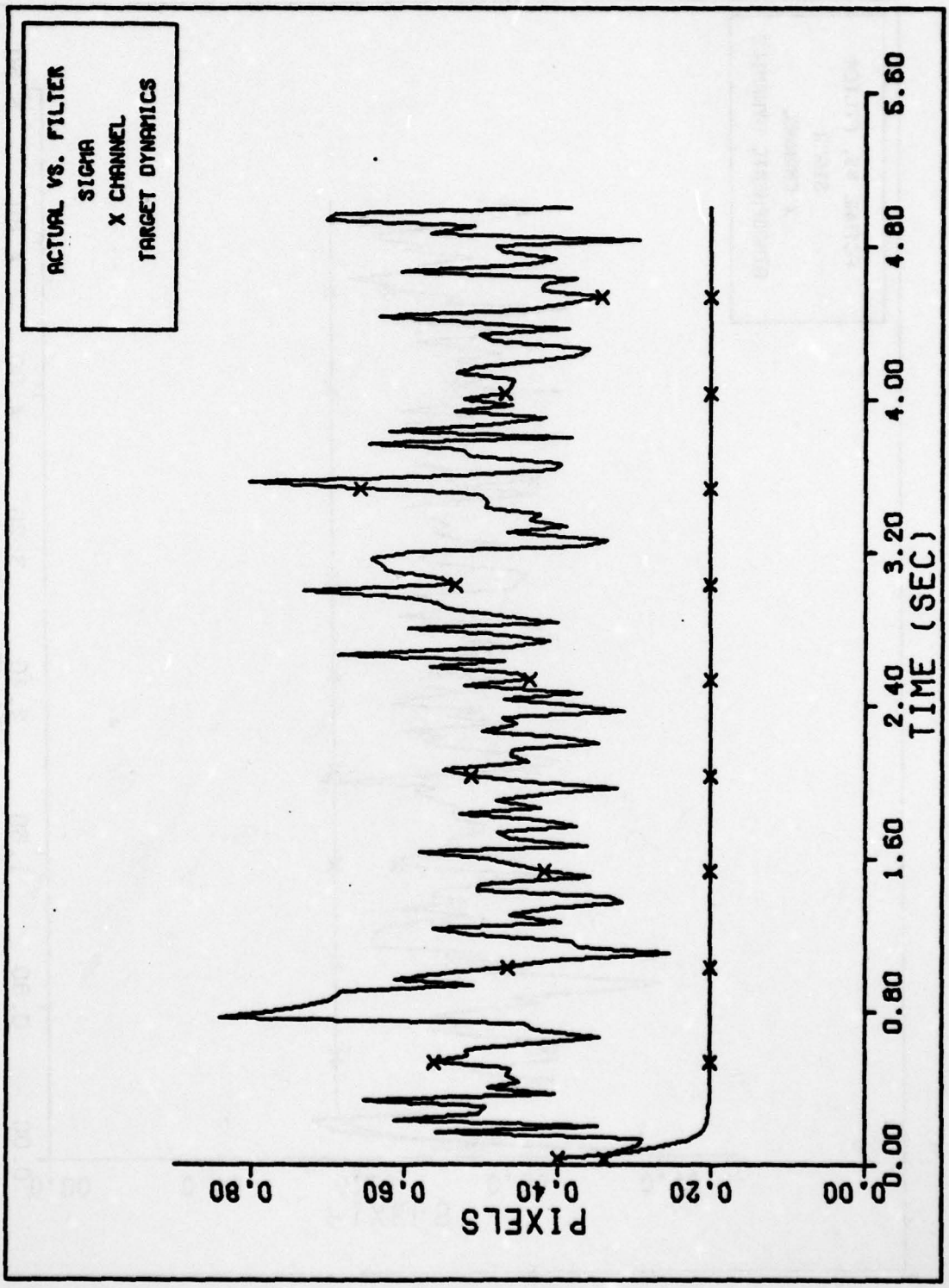
X CHANNEL DYNAMICS ERROR (S/N= 10)

Figure 19b. Case 19 Performance Plot



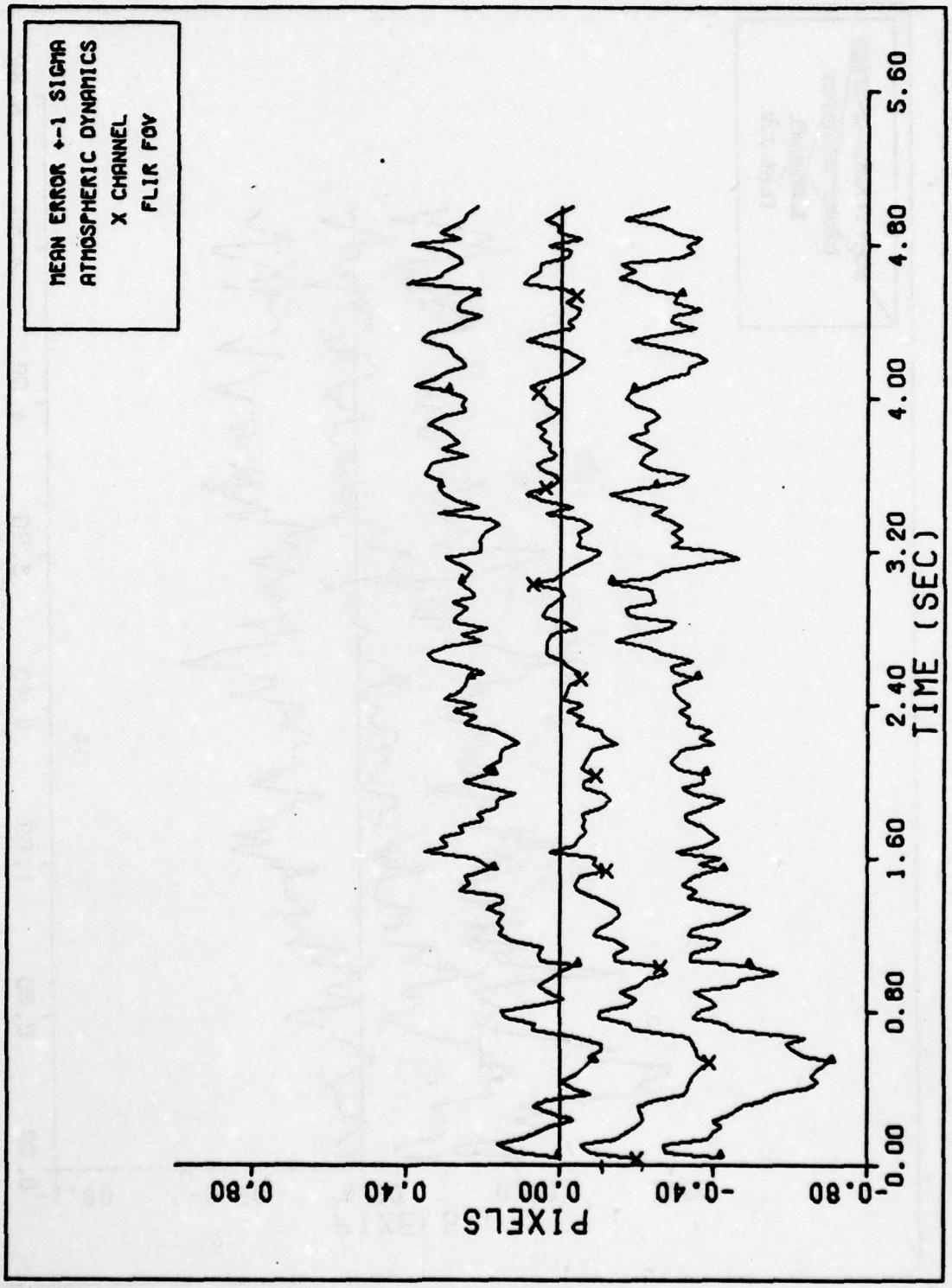
FILTER VS. ACTUAL SIGMA PLOT (S/N = 10)

Figure 19c. Case 19 Performance Plot



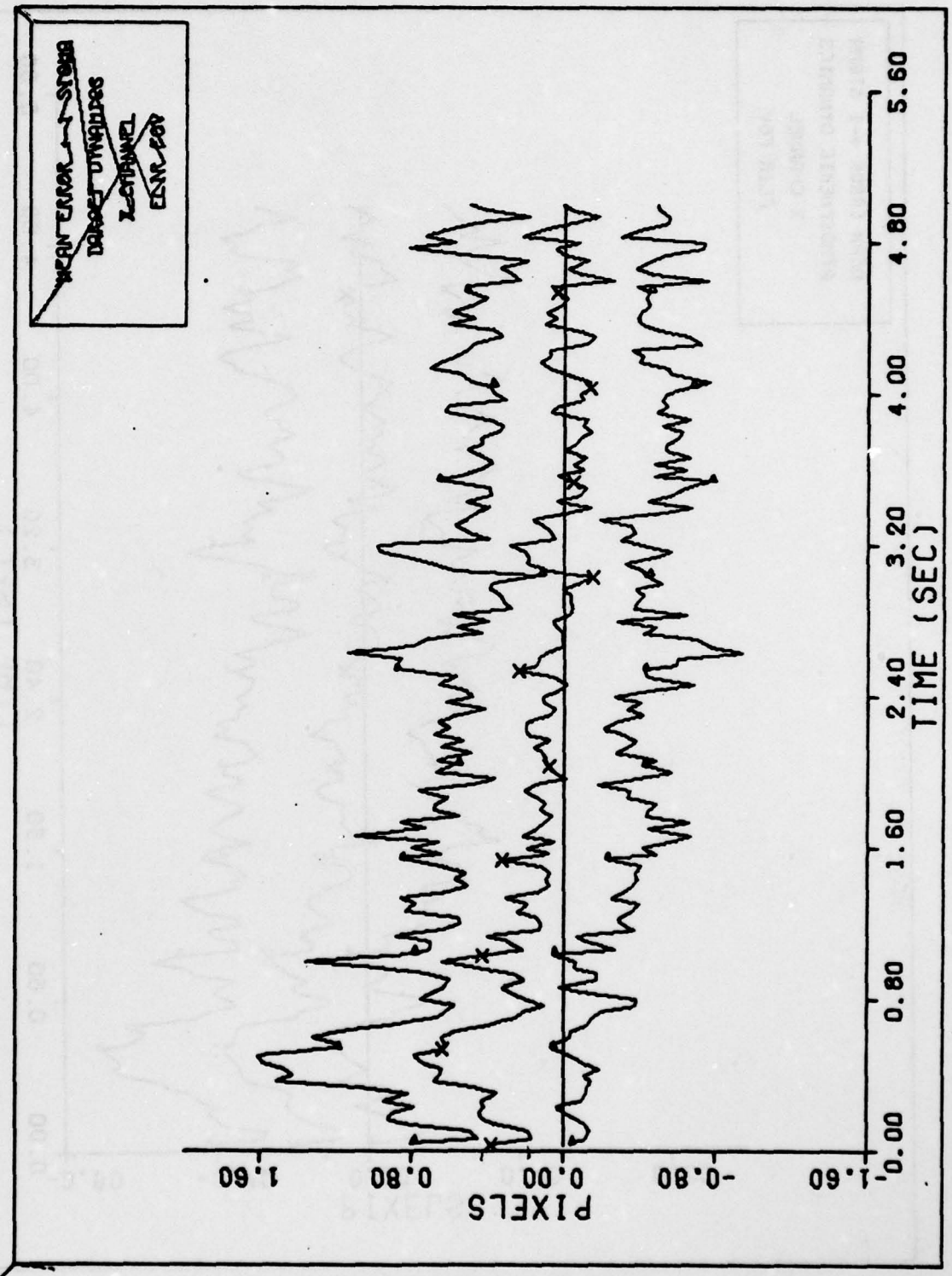
FILTER VS. ACTUAL SIGMA PLOT (S/N = 10)

Figure 19d. Case 19 Performance Plot



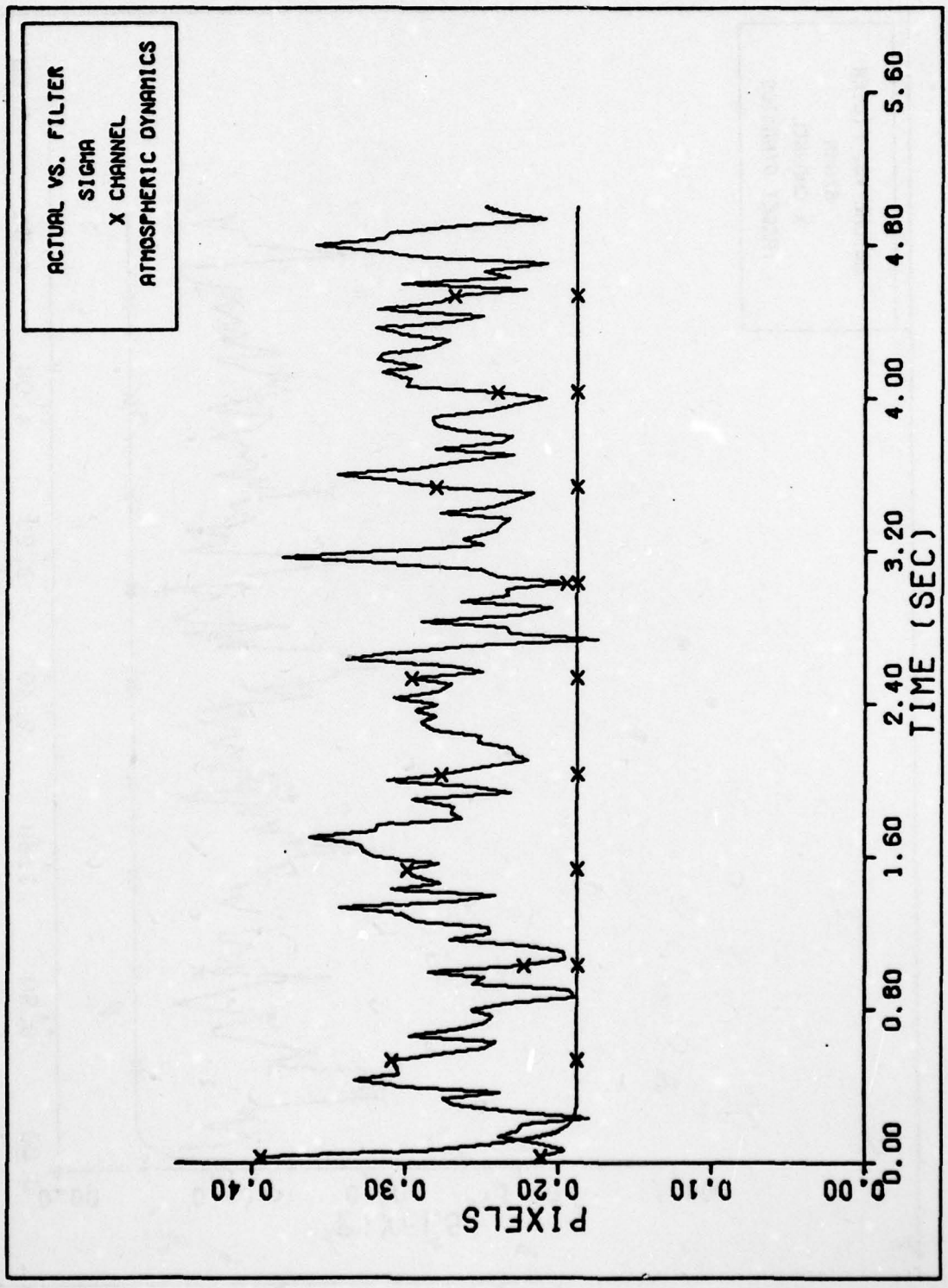
X CHANNEL ATMOSPHERICS ERROR (S/N=10)

Figure 20a. Case 20 Performance Plot



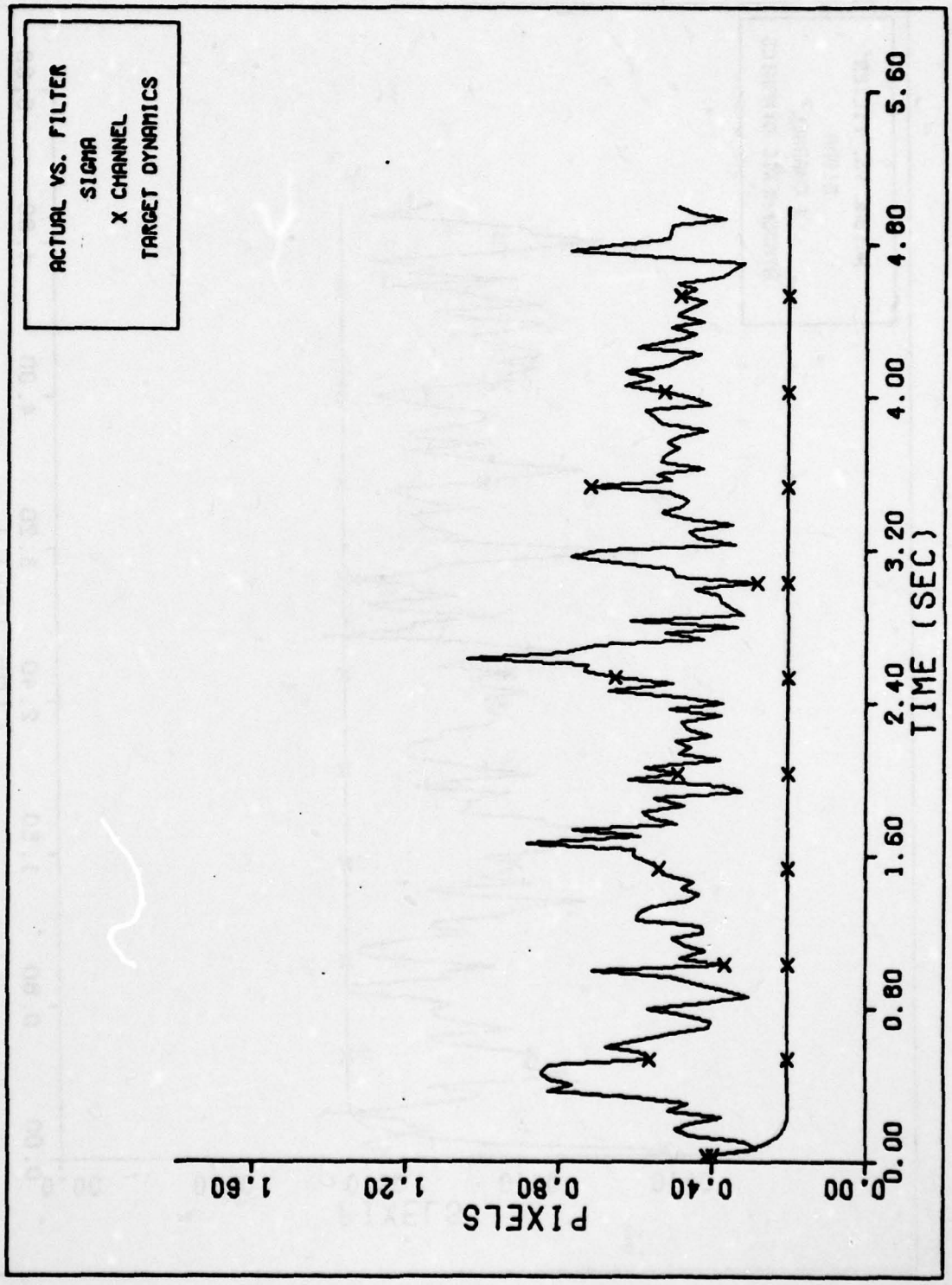
X CHANNEL DYNAMICS ERROR (S/N=10)

Figure 20b. Case 20 Performance Plot



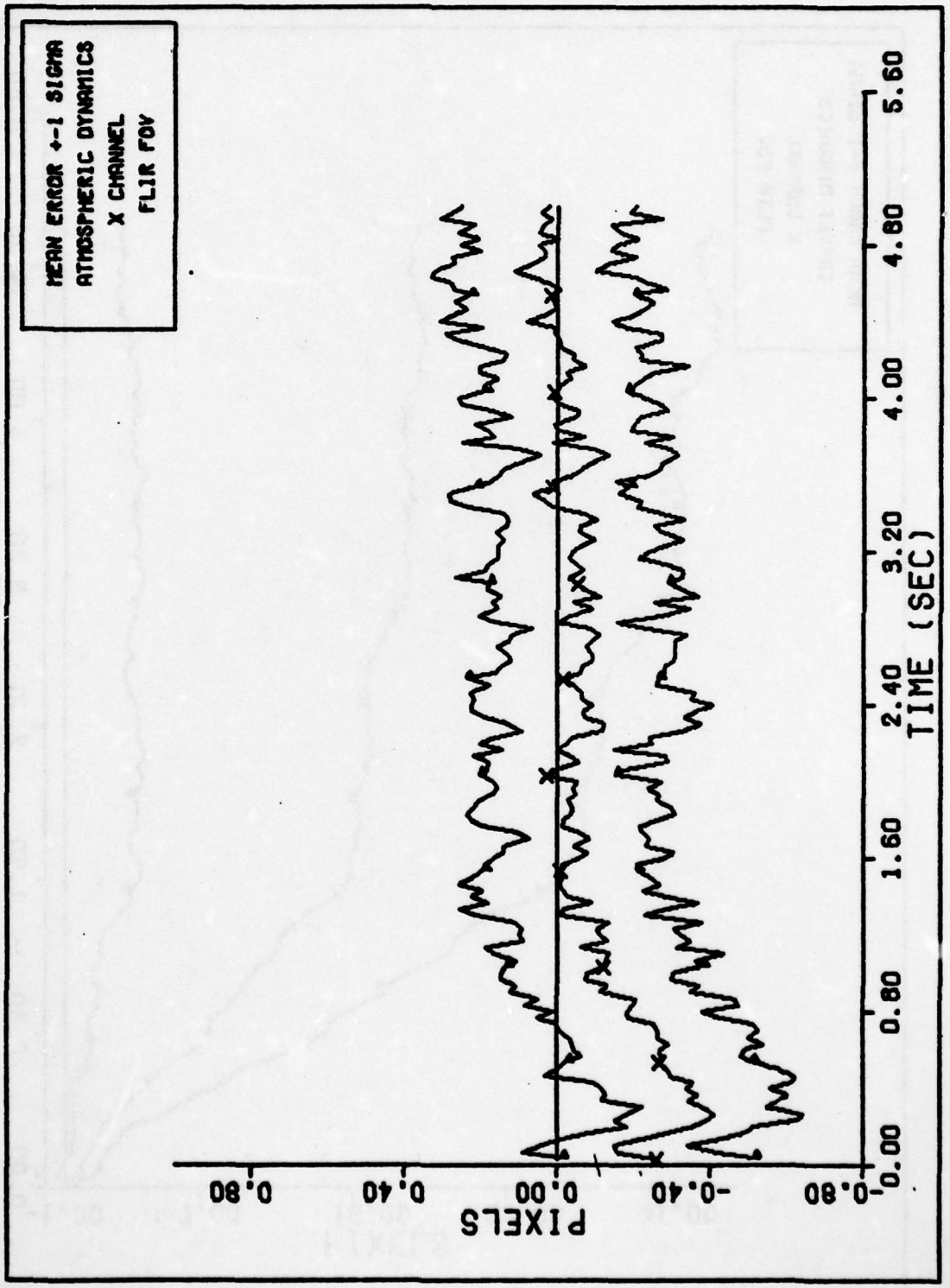
FILTER VS. ACTUAL SIGMA PLOT (S/N = 10)

Figure 20c. Case 20 Performance Plot



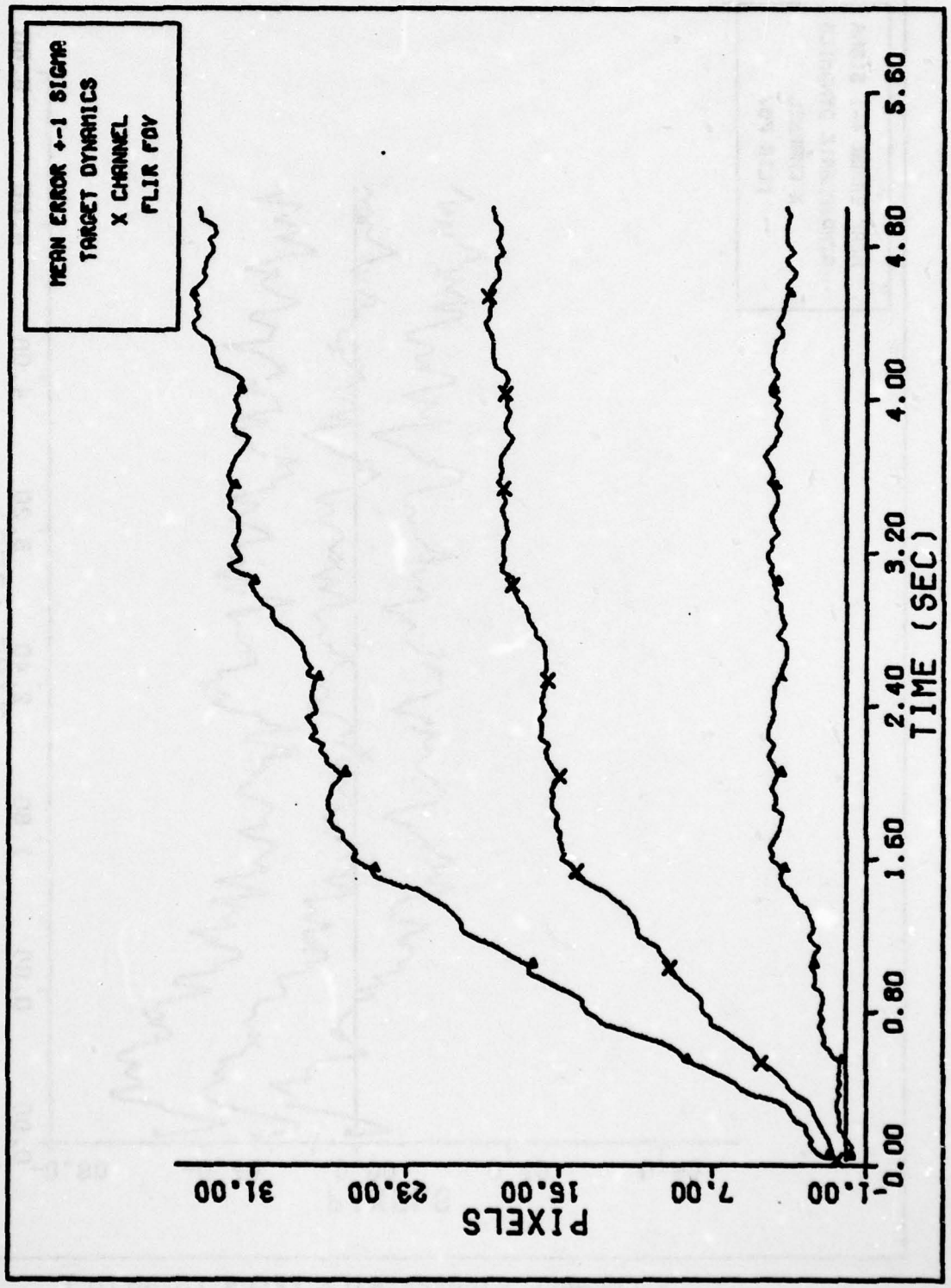
FILTER VS. ACTUAL SIGMA PLOT (S/N = 10)

Figure 20d. Case 20 Performance Plot



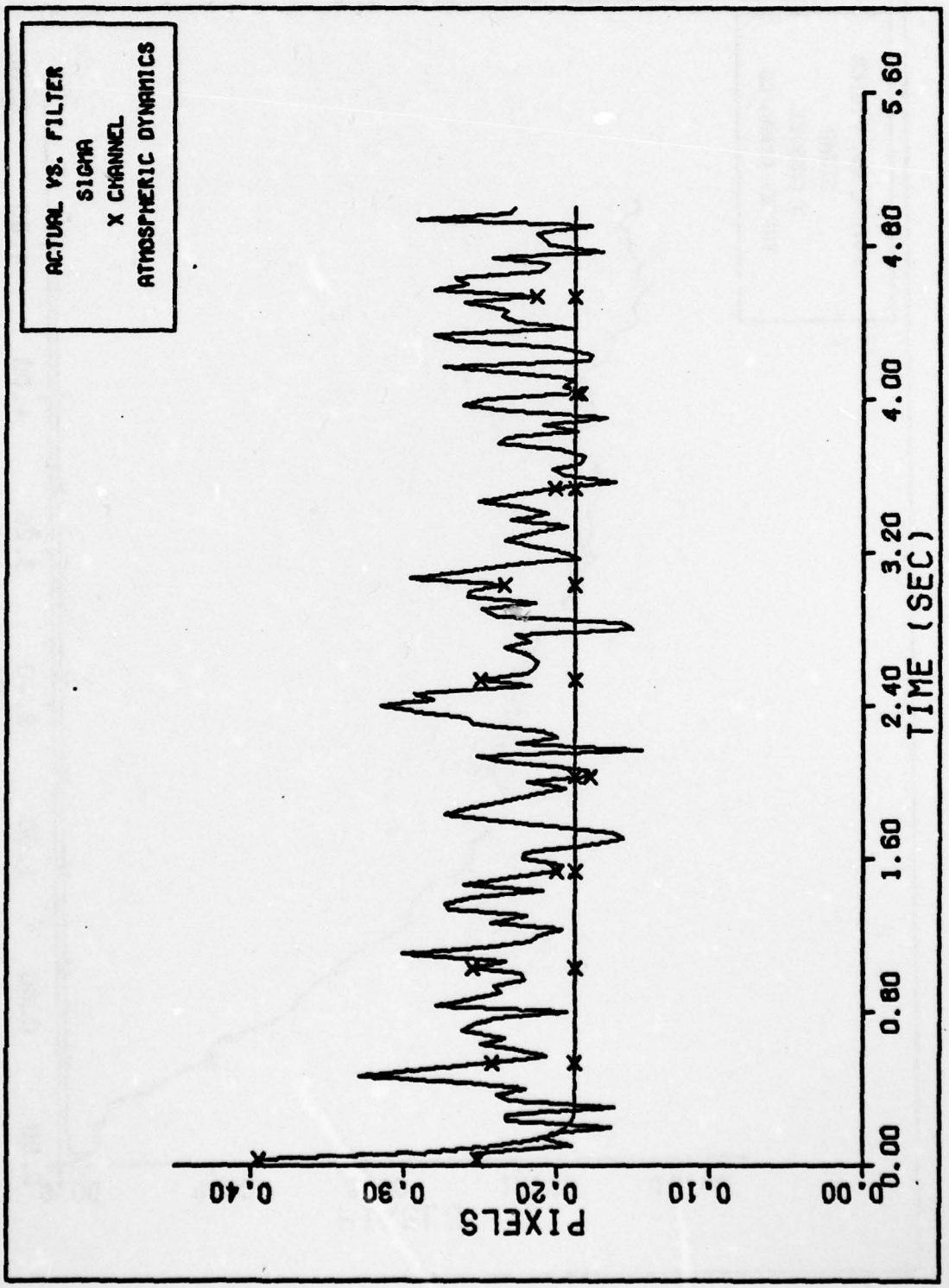
X CHANNEL ATMOSPHERICS ERROR (S/N= 10)

Figure 21a. Case 21 Performance Plot



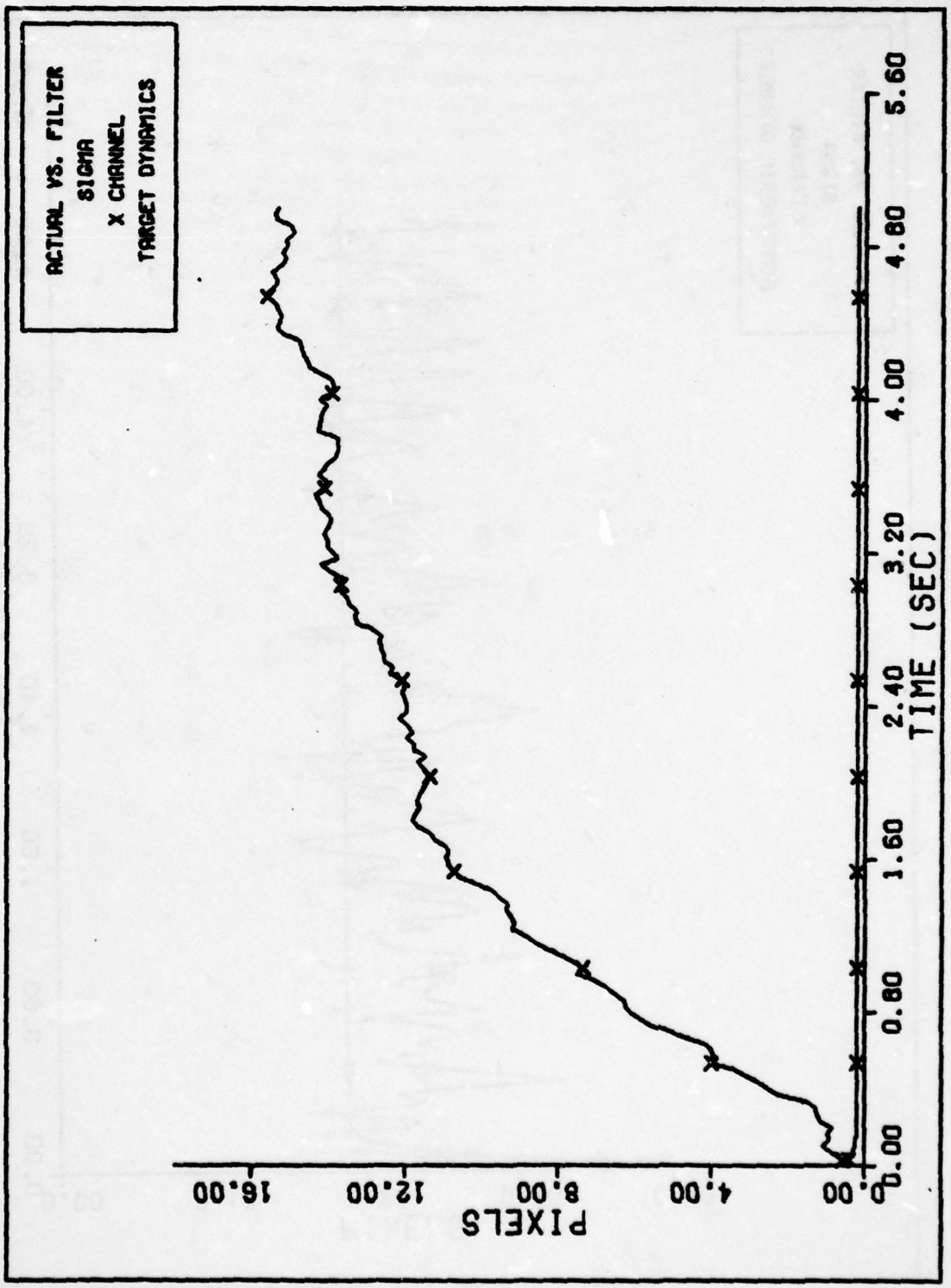
X CHANNEL DYNAMICS ERROR (S/N= 10)

Figure 21b. Case 21 Performance Plot



FILTER VS. ACTUAL SIGMA PLOT (S/N = 10)

Figure 21c. Case 21 Performance Plot



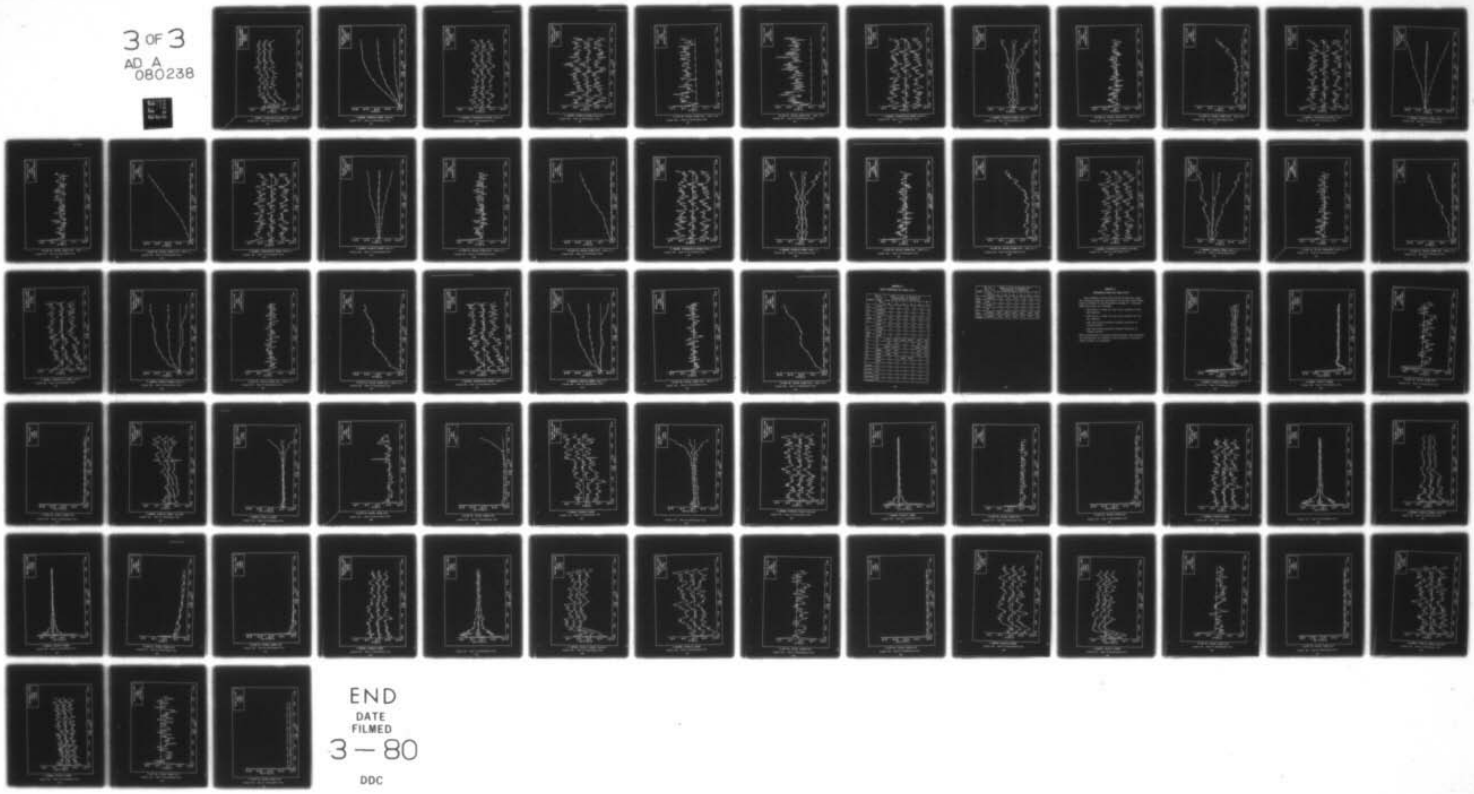
FILTER VS. ACTUAL SIGMA PLOT (S/N = 10)

Figure 21d. Case 21 Performance Plot

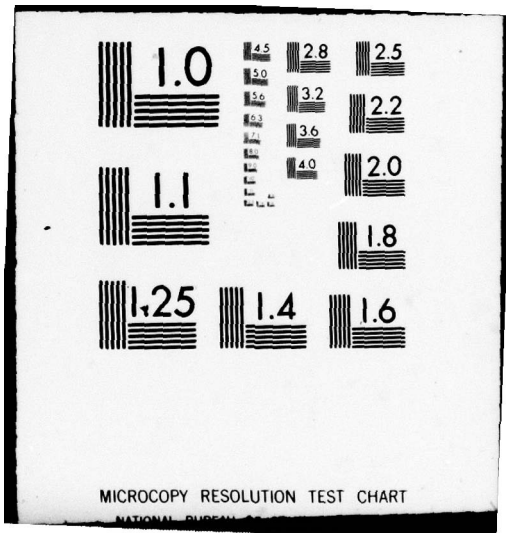
AD-A080 238 AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH SCHOO--ETC F/G 17/5
AN ADAPTIVE DISTRIBUTED-MEASUREMENT EXTENDED KALMAN FILTER FOR --ETC(U)
DEC 79 R L JENSEN, D A HARNLY

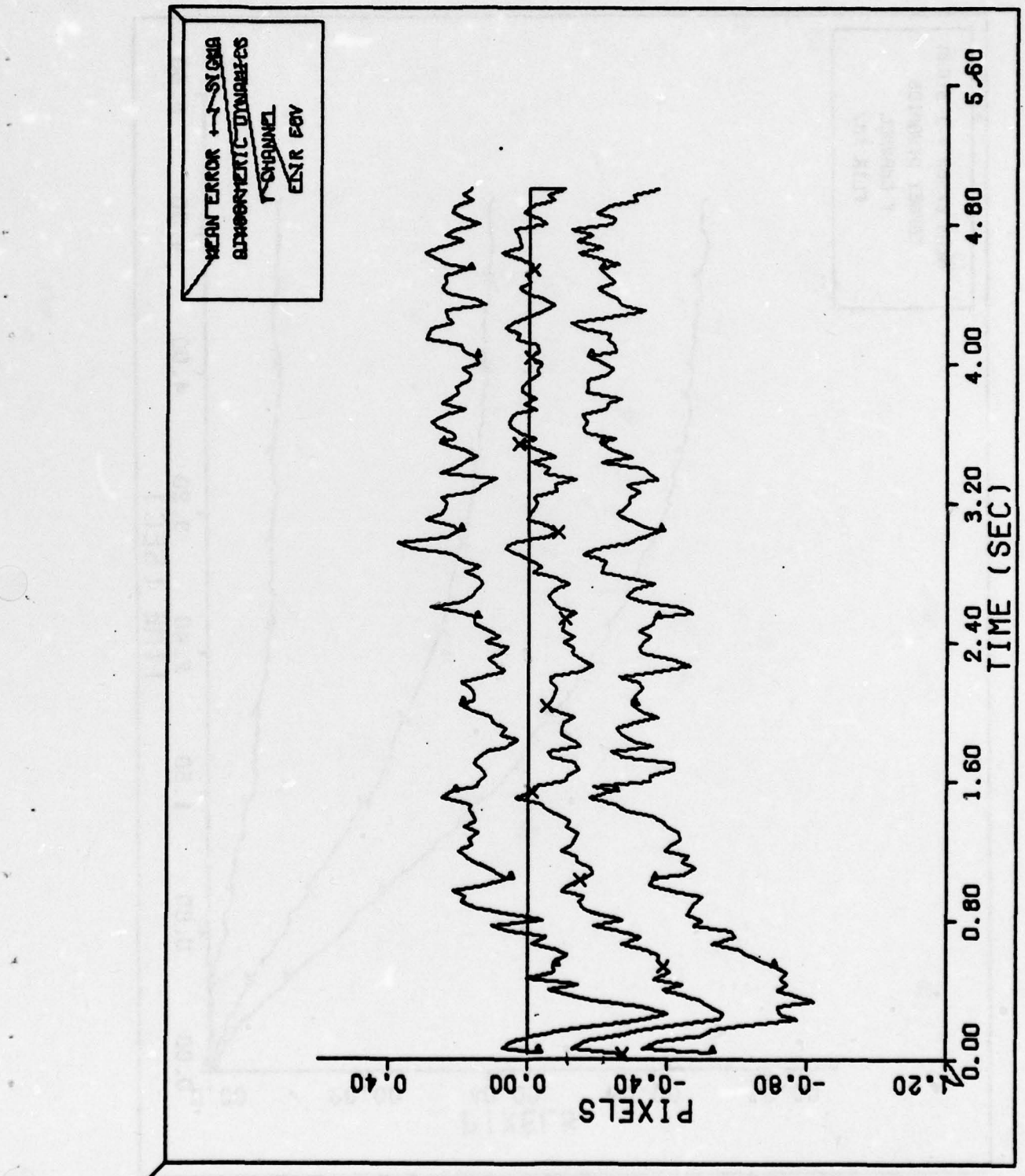
UNCLASSIFIED AFIT/GA/EE/79-1-VOL-2 NL

3 OF 3
AD A
080238



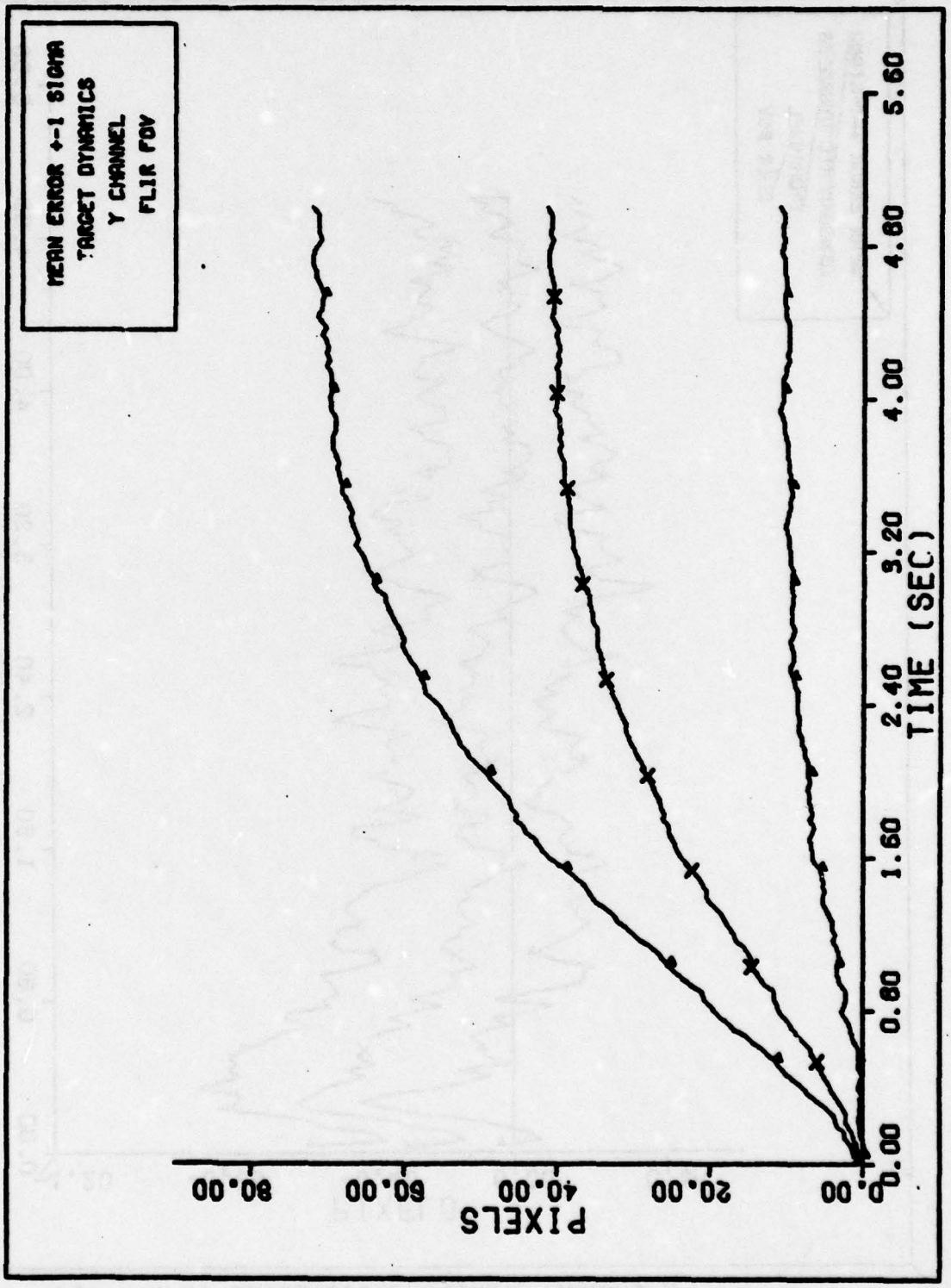
END
DATE
FILMED
3-80
DDC


1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0
4.5
5.0
5.6
6.3
MICROCOPY RESOLUTION TEST CHART



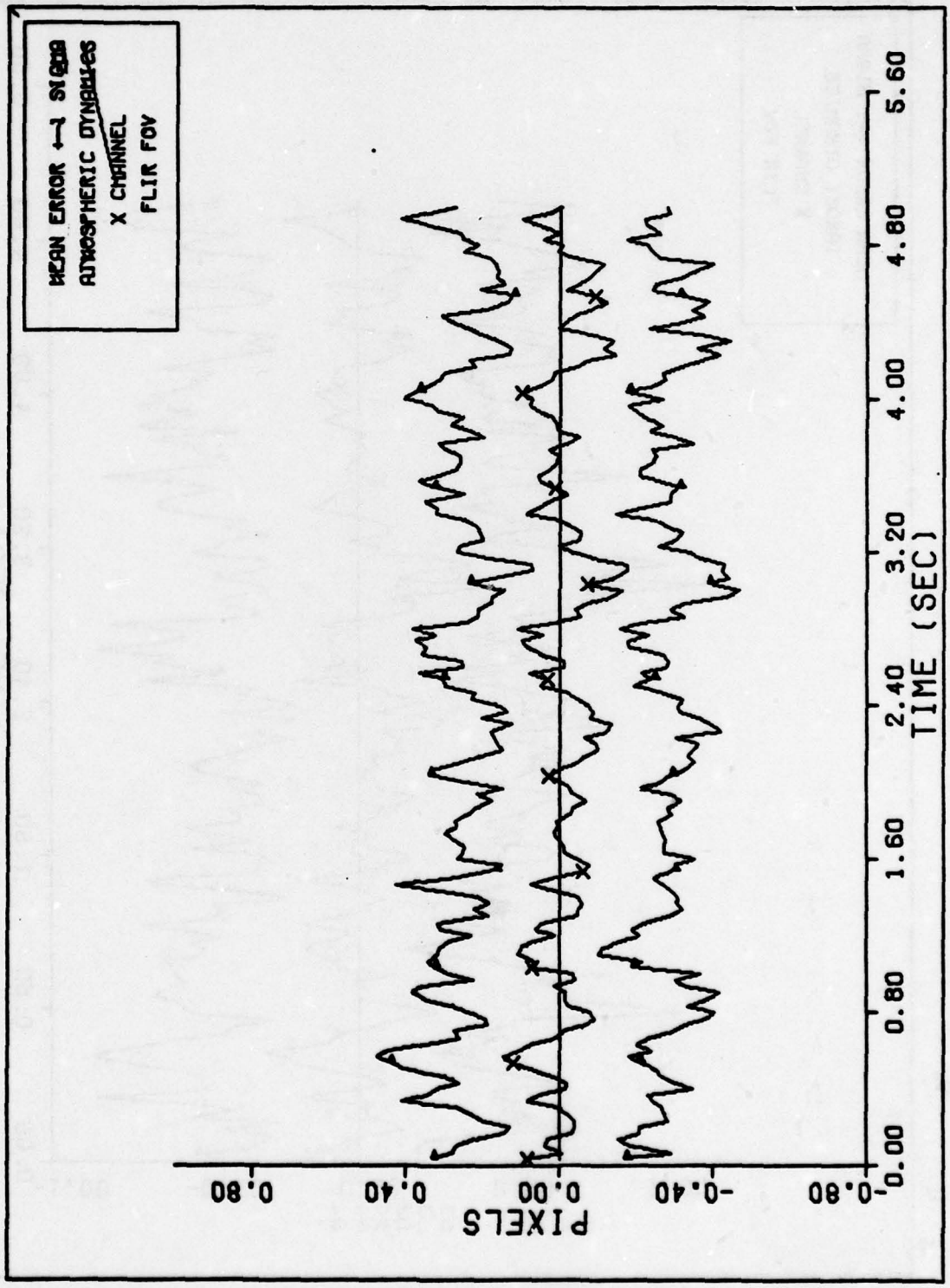
Y CHANNEL ATMOSPHERICS ERROR (S/N = 10)

Figure 21e. Case 21 Performance Plot



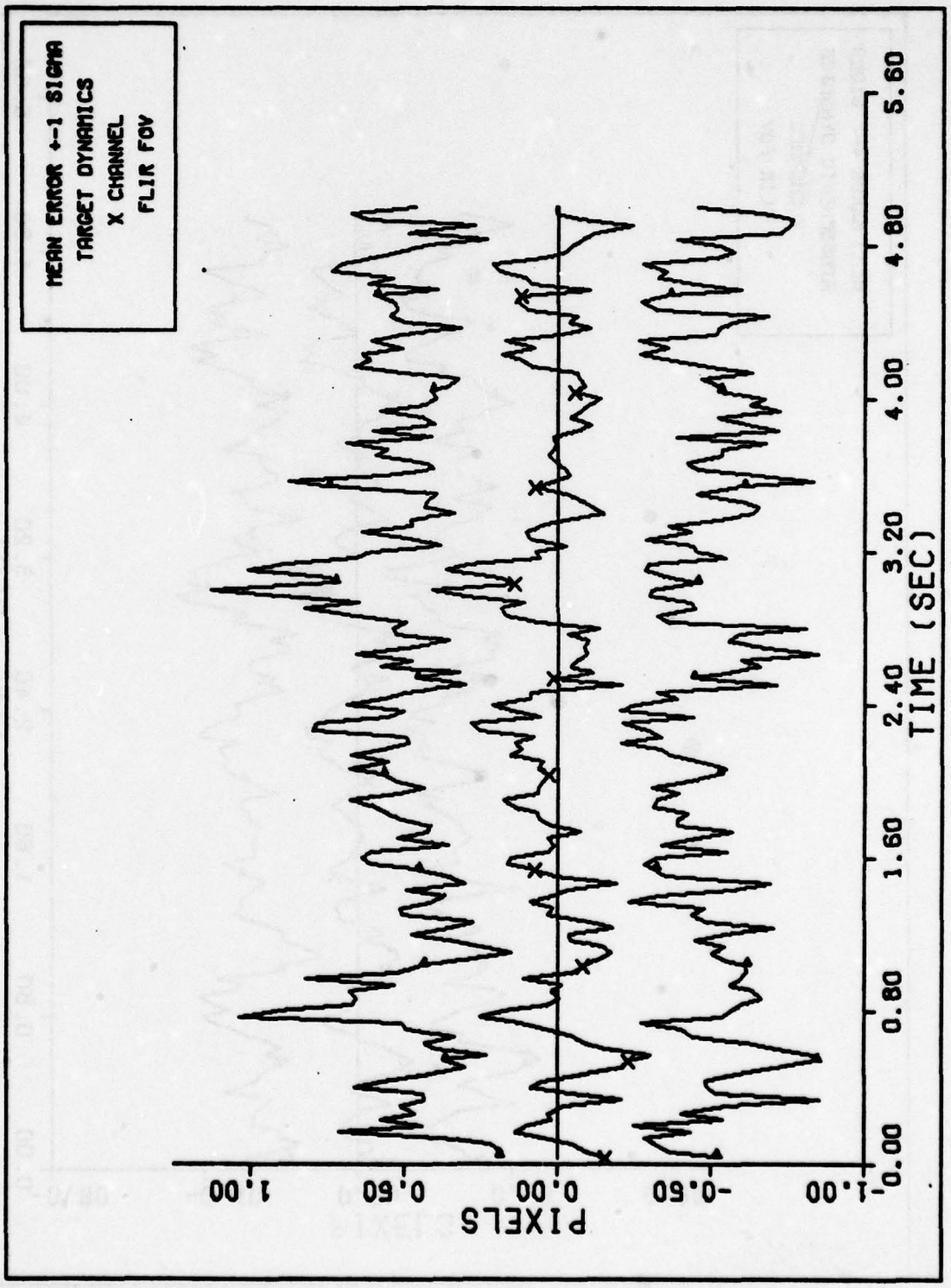
Y CHANNEL DYNAMICS ERROR (S/N=10)

Figure 21f. Case 21 Performance Plot



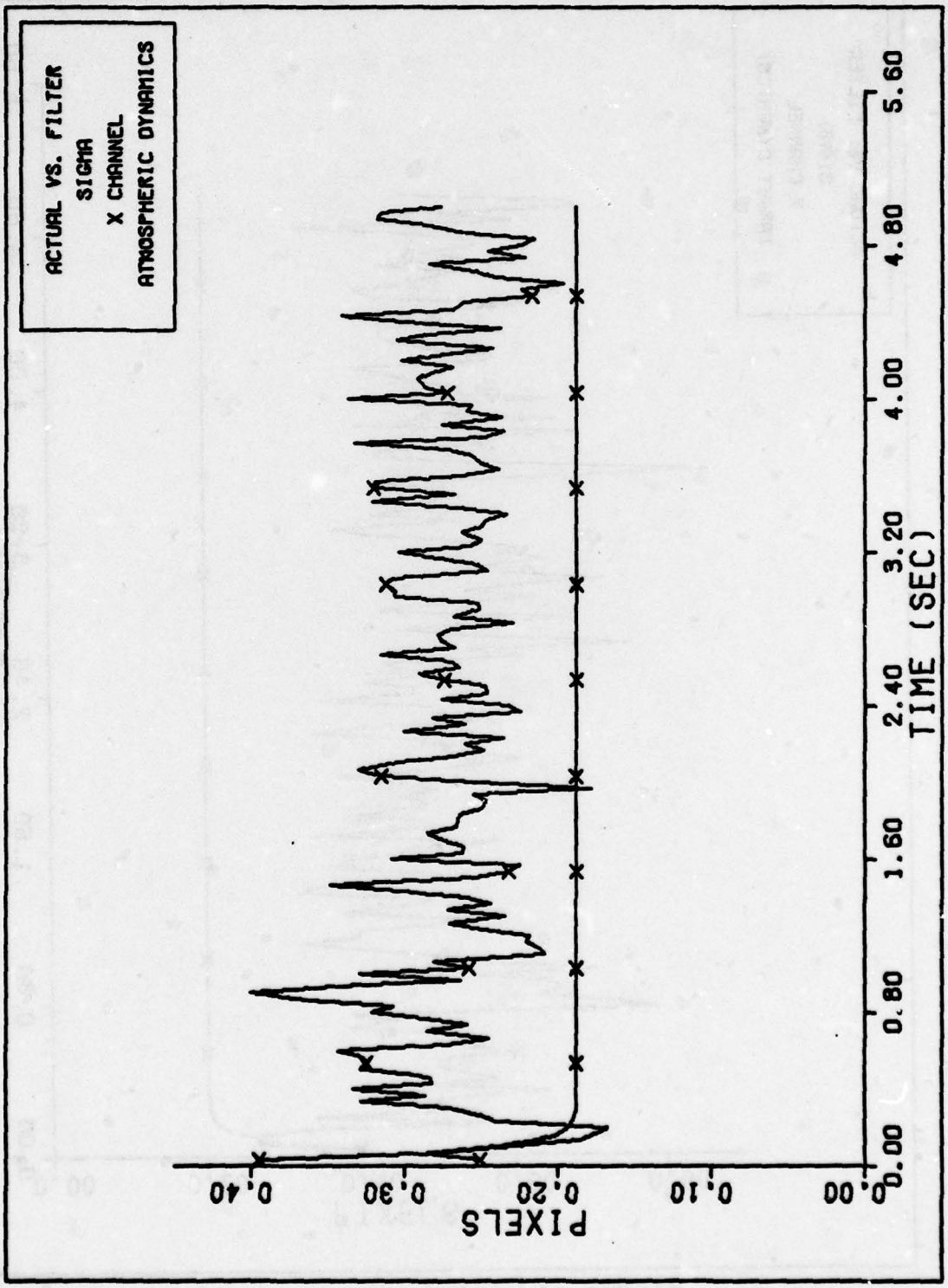
X CHANNEL ATMOSPHERICS ERROR (S/N= 10)

Figure 22a. Case 23 Performance Plot



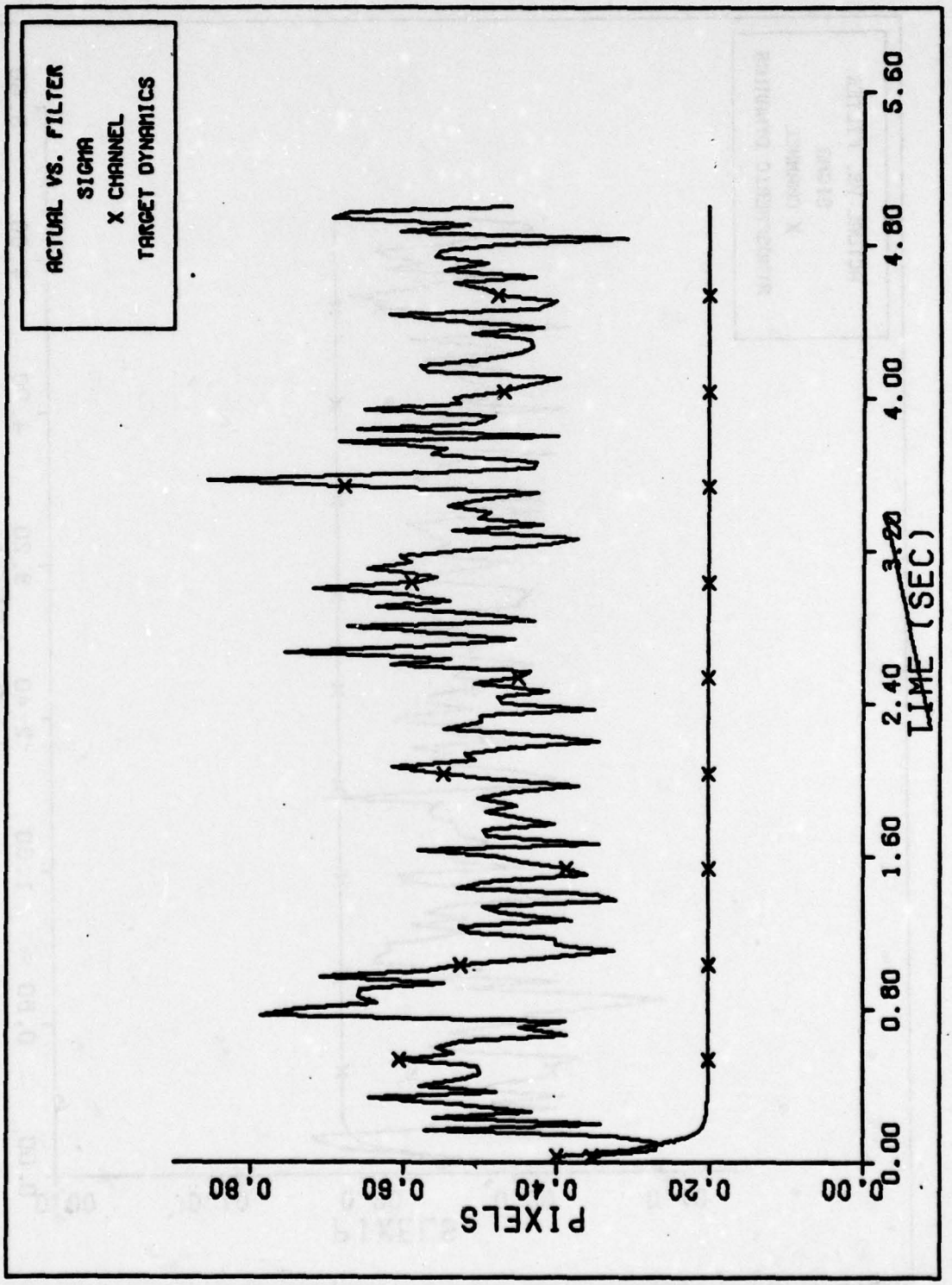
X CHANNEL DYNAMICS ERROR (S/N= 10)

Figure 22b. Case 23 Performance Plot



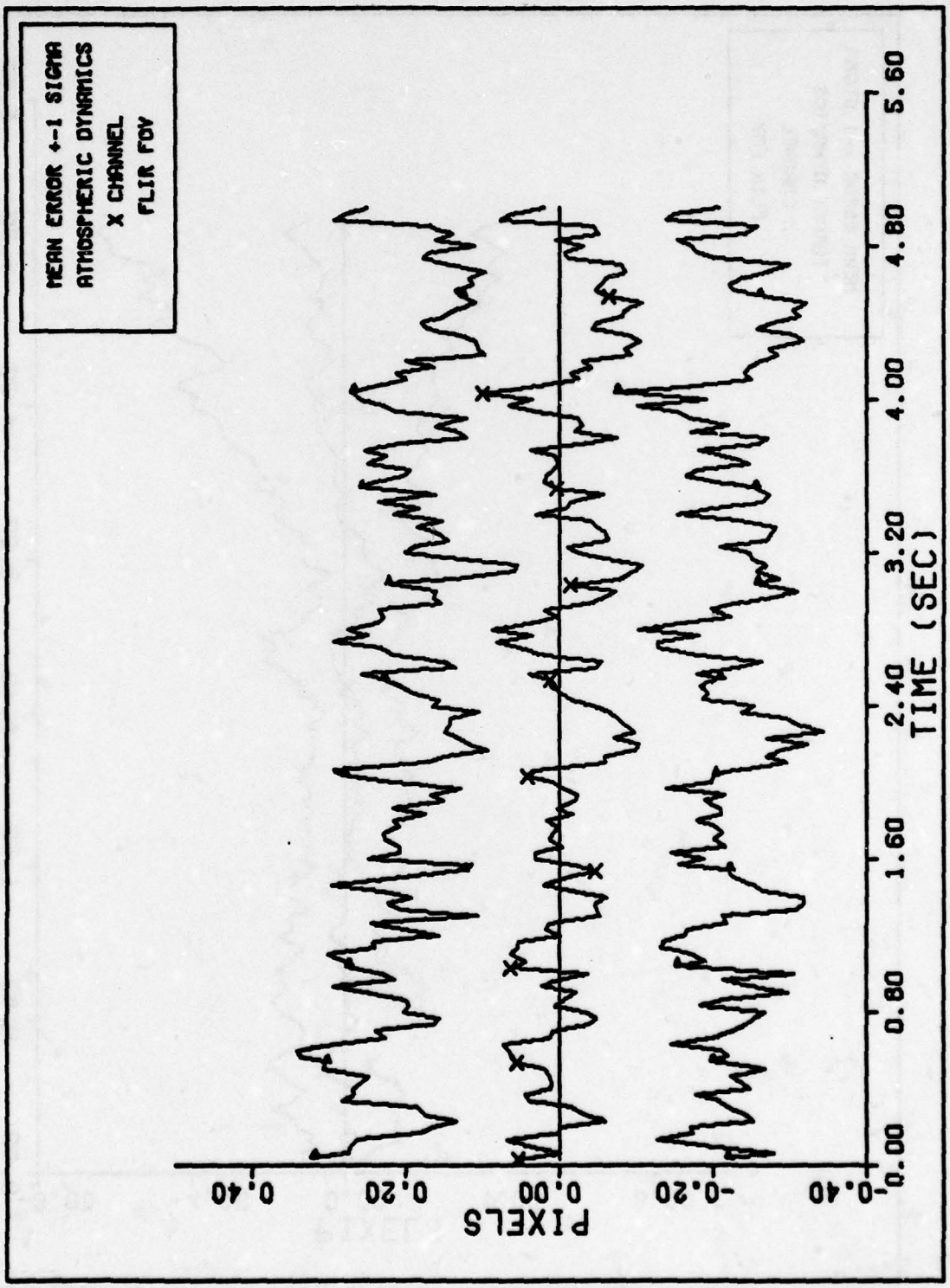
FILTER VS. ACTUAL SIGMA PLOT (S/N = 10)

Figure 22c. Case 23 Performance Plot



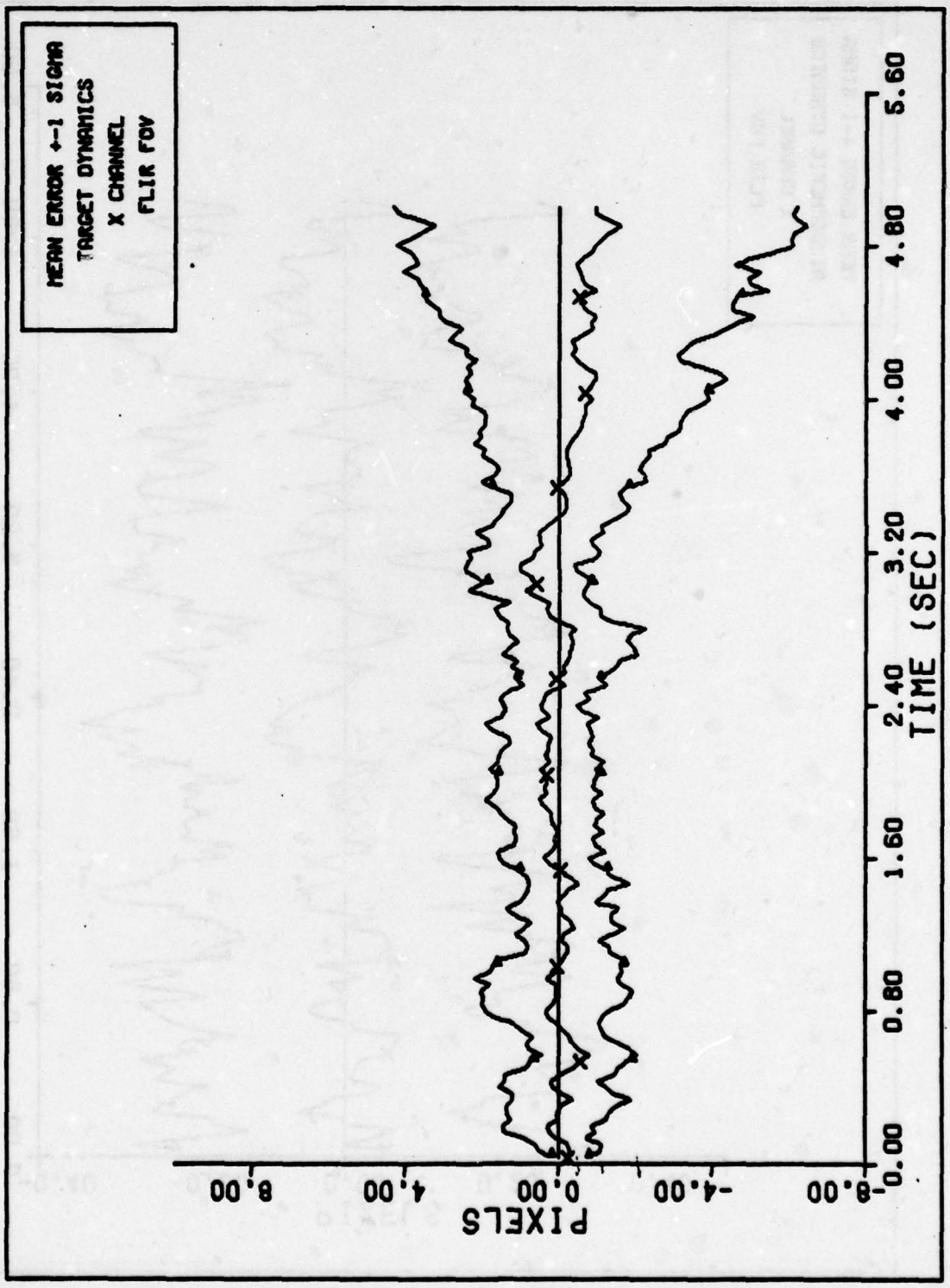
FILTER VS. ACTUAL SIGMA PLOT (S/N = 10)

Figure 22d. Case 23 Performance Plot



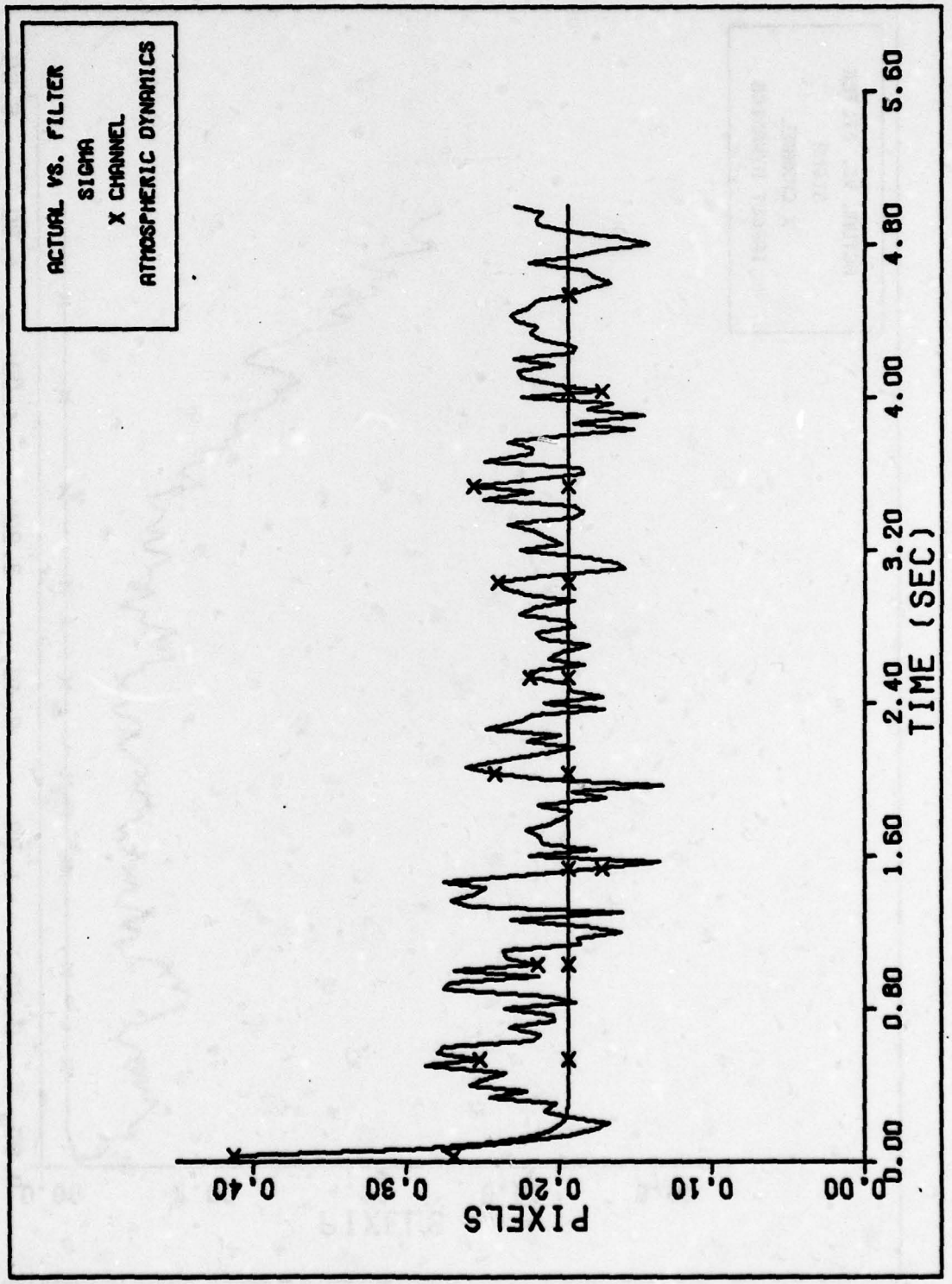
X CHANNEL ATMOSPHERICS ERROR (S/N= 10)

Figure 23a. Case 25 Performance Plot



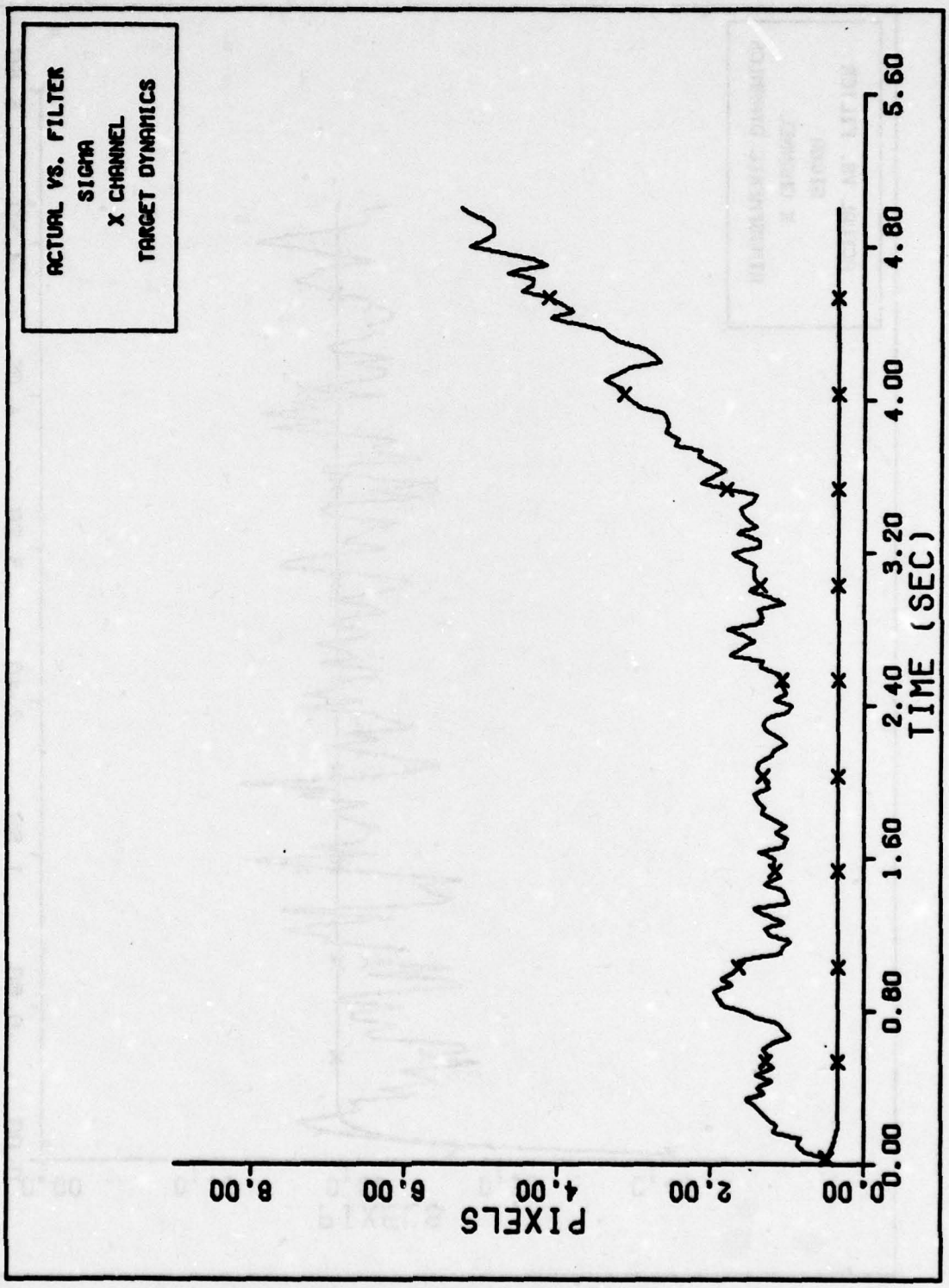
X CHANNEL DYNAMICS ERROR (S/N= 10)

Figure 23b. Case 25 Performance Plot



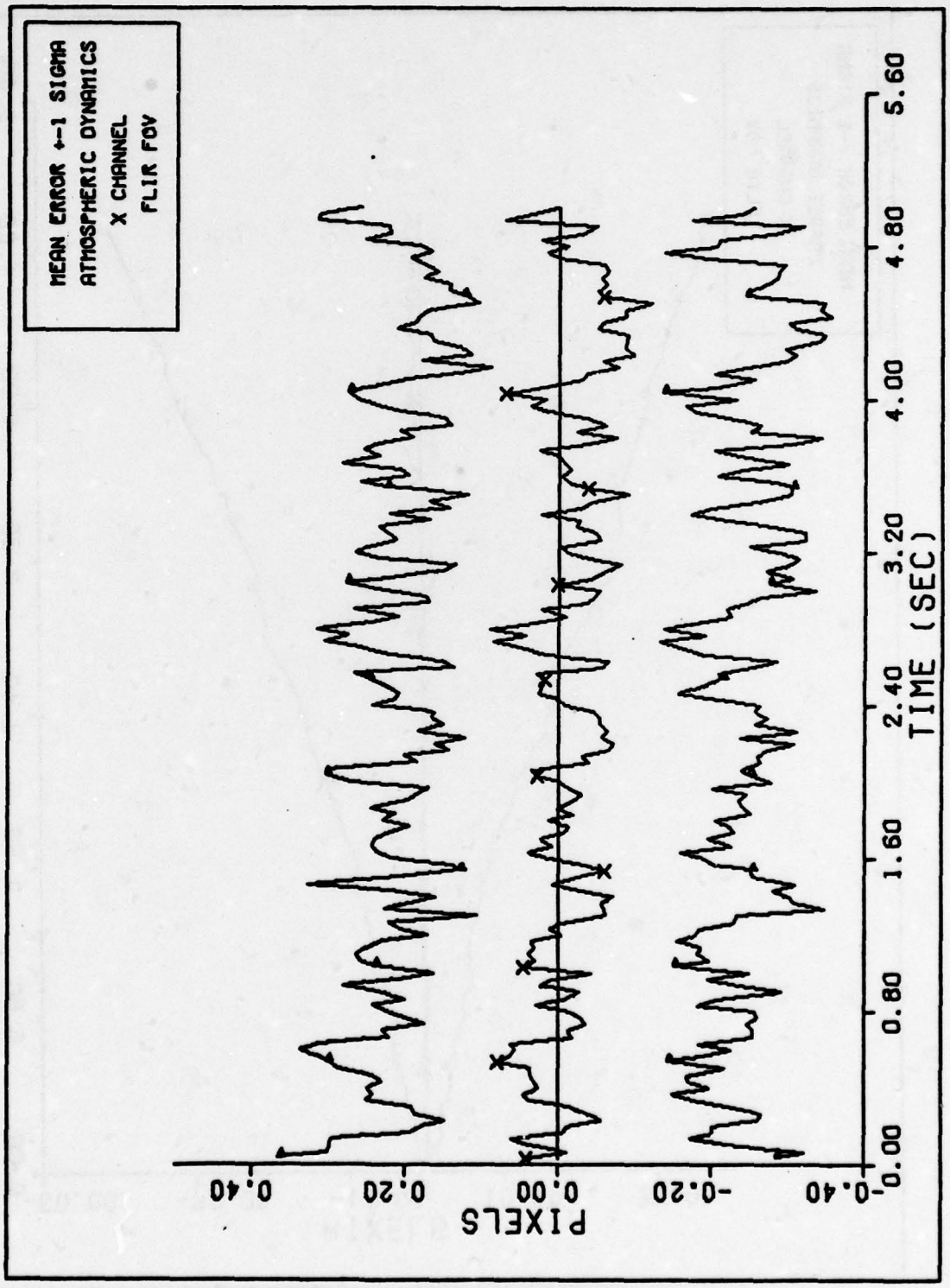
FILTER VS. ACTUAL SIGMA PLOT (S/N = 10)

Figure 23c. Case 25 Performance Plot



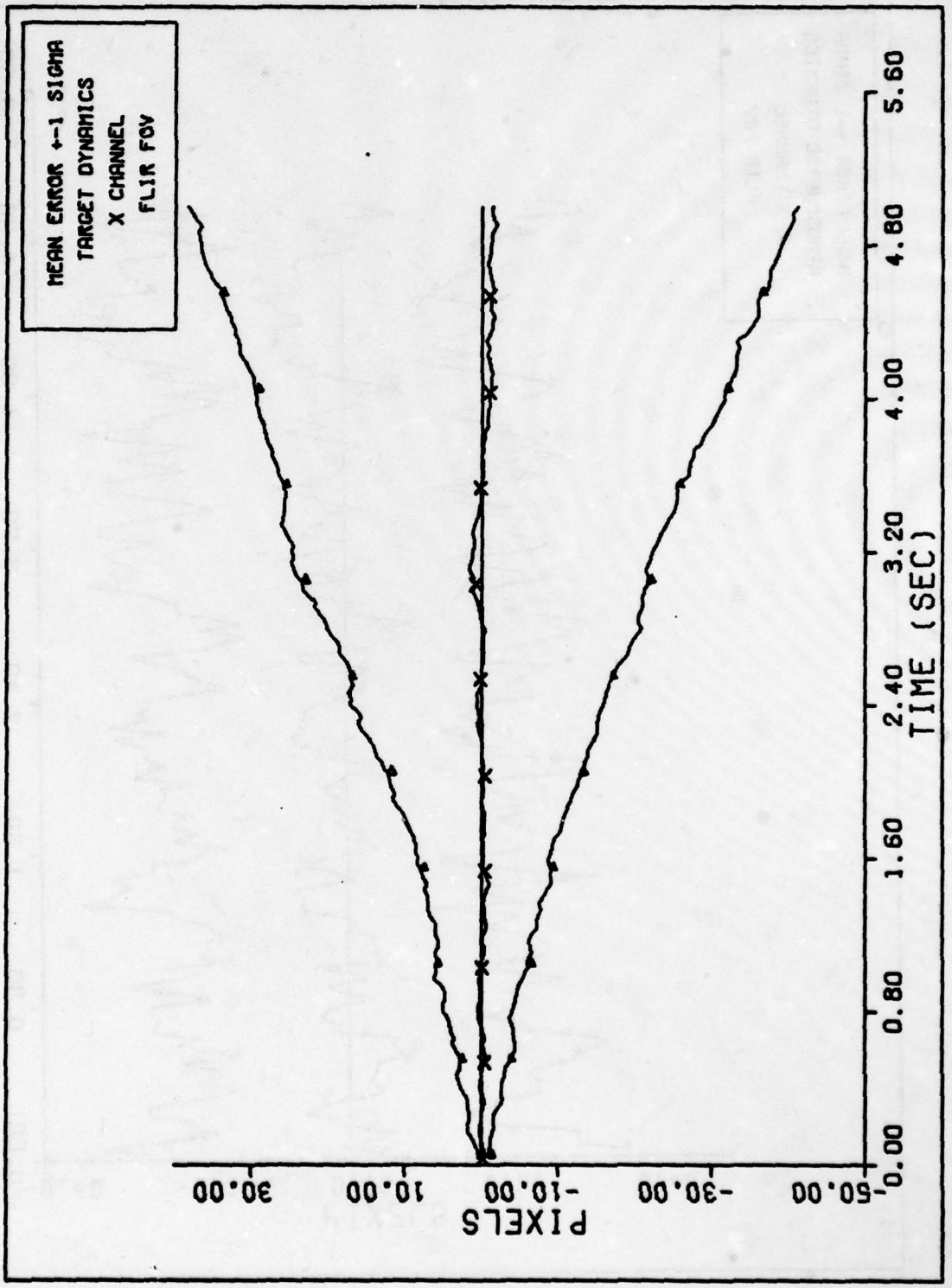
FILTER VS. ACTUAL SIGMA PLOT (S/N = 10)

Figure 23d. Case 25 Performance Plot



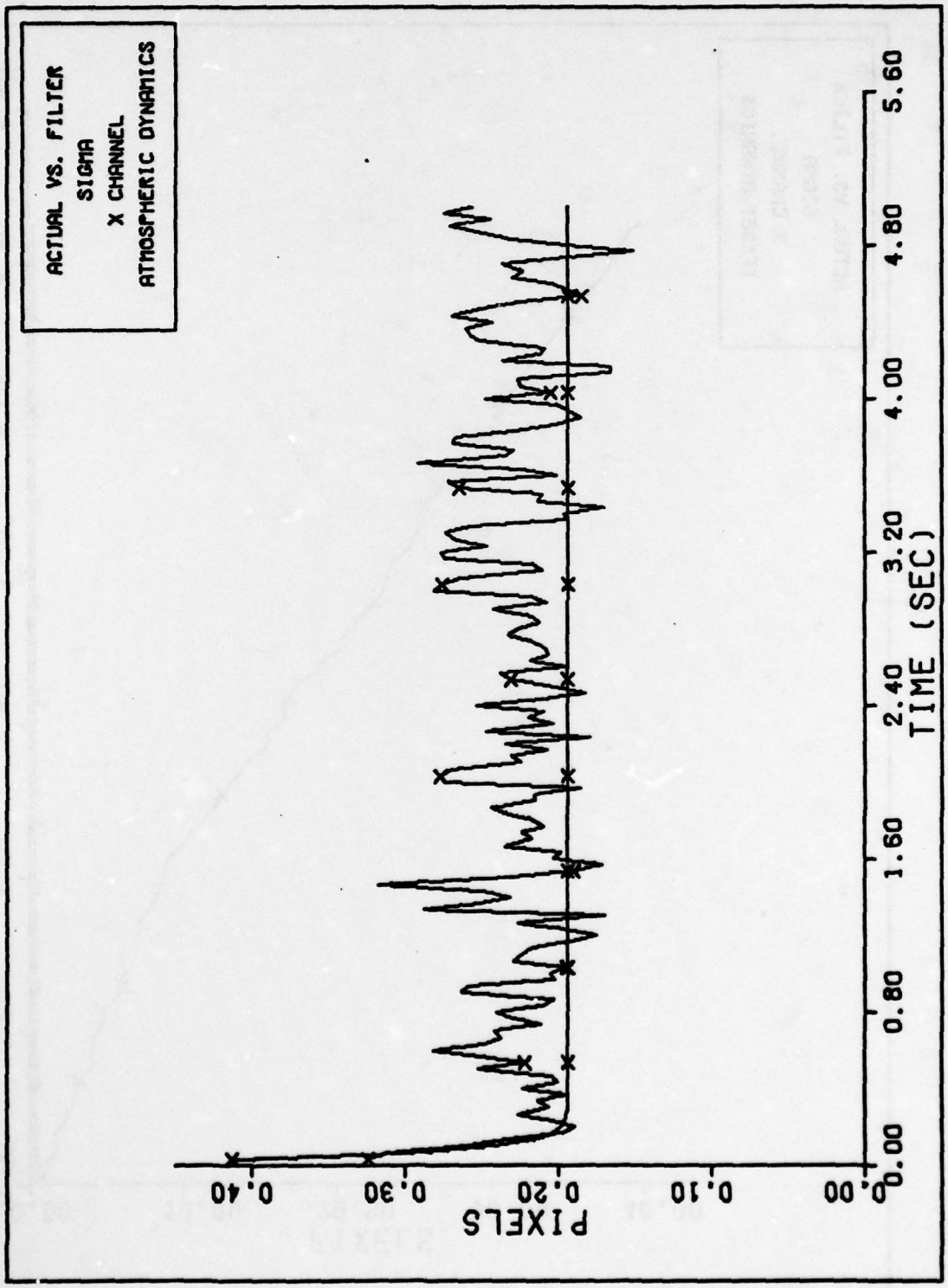
X CHANNEL ATMOSPHERICS ERROR (S/N= /)

Figure 24a. Case 26 Performance Plot



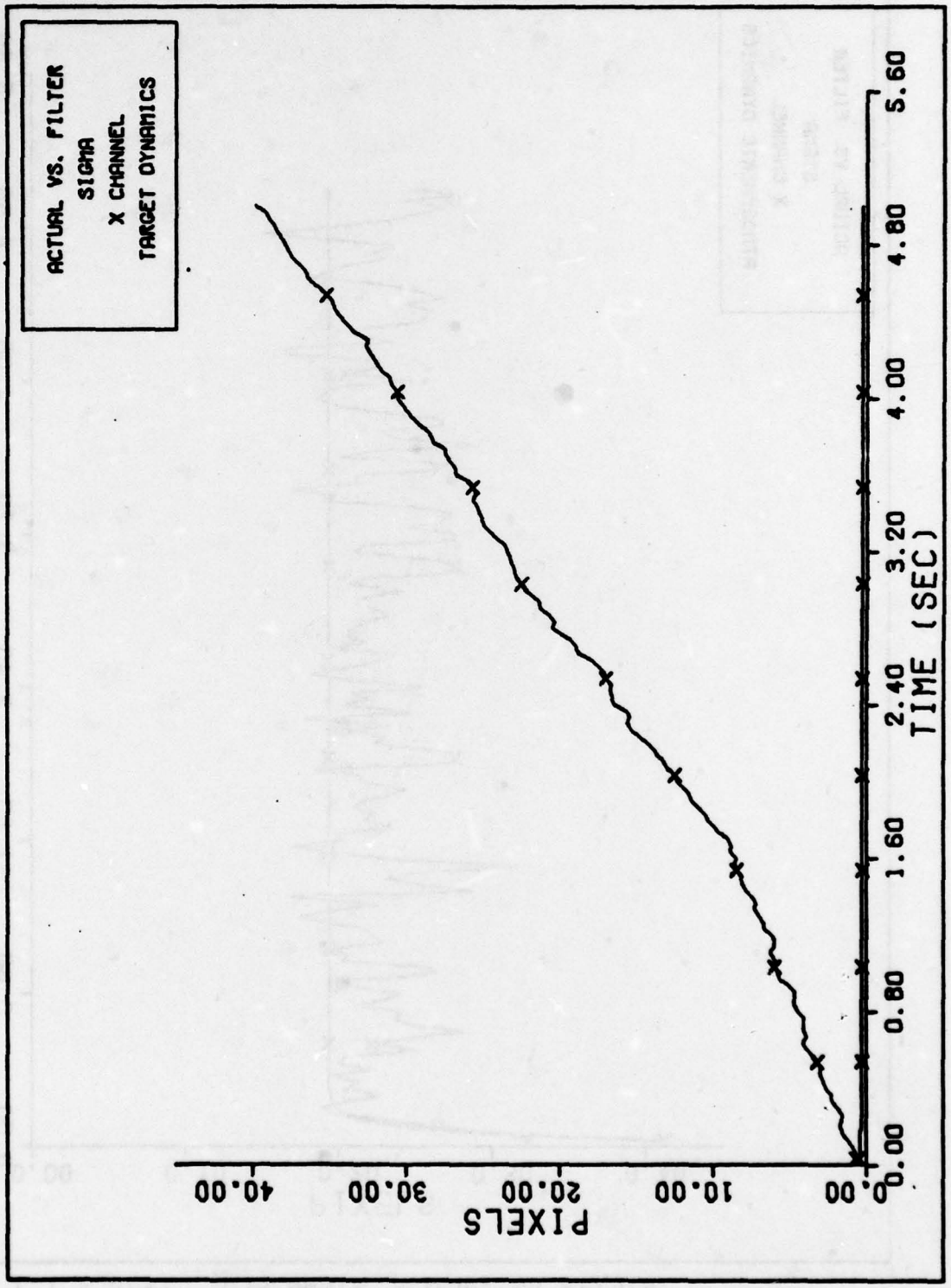
X CHANNEL DYNAMICS ERROR (S/N= /)

Figure 24b. Case 26 Performance Plot



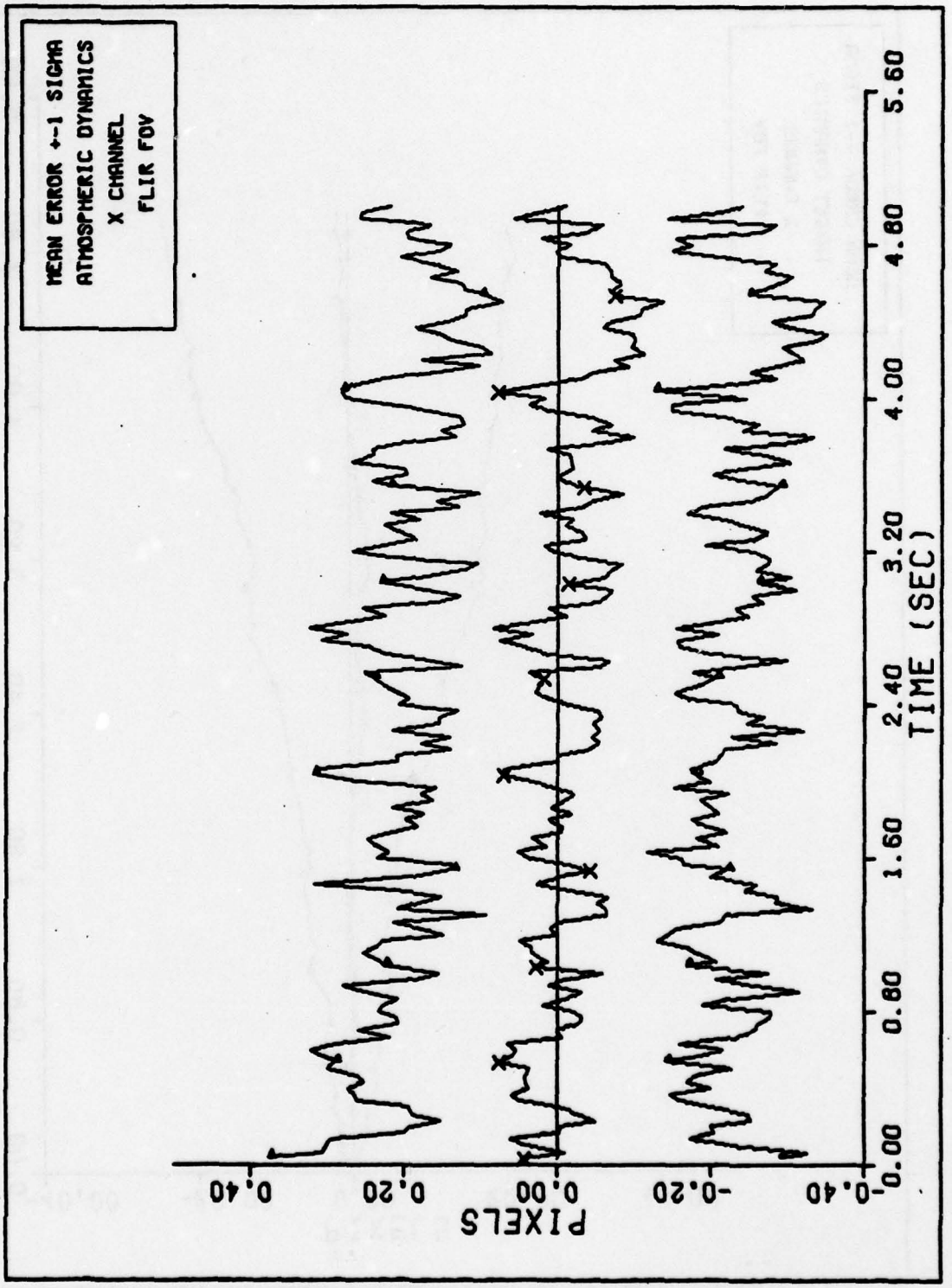
FILTER VS. ACTUAL SIGMA PLOT (S/N = /)

Figure 24c. Case 26 Performance Plot



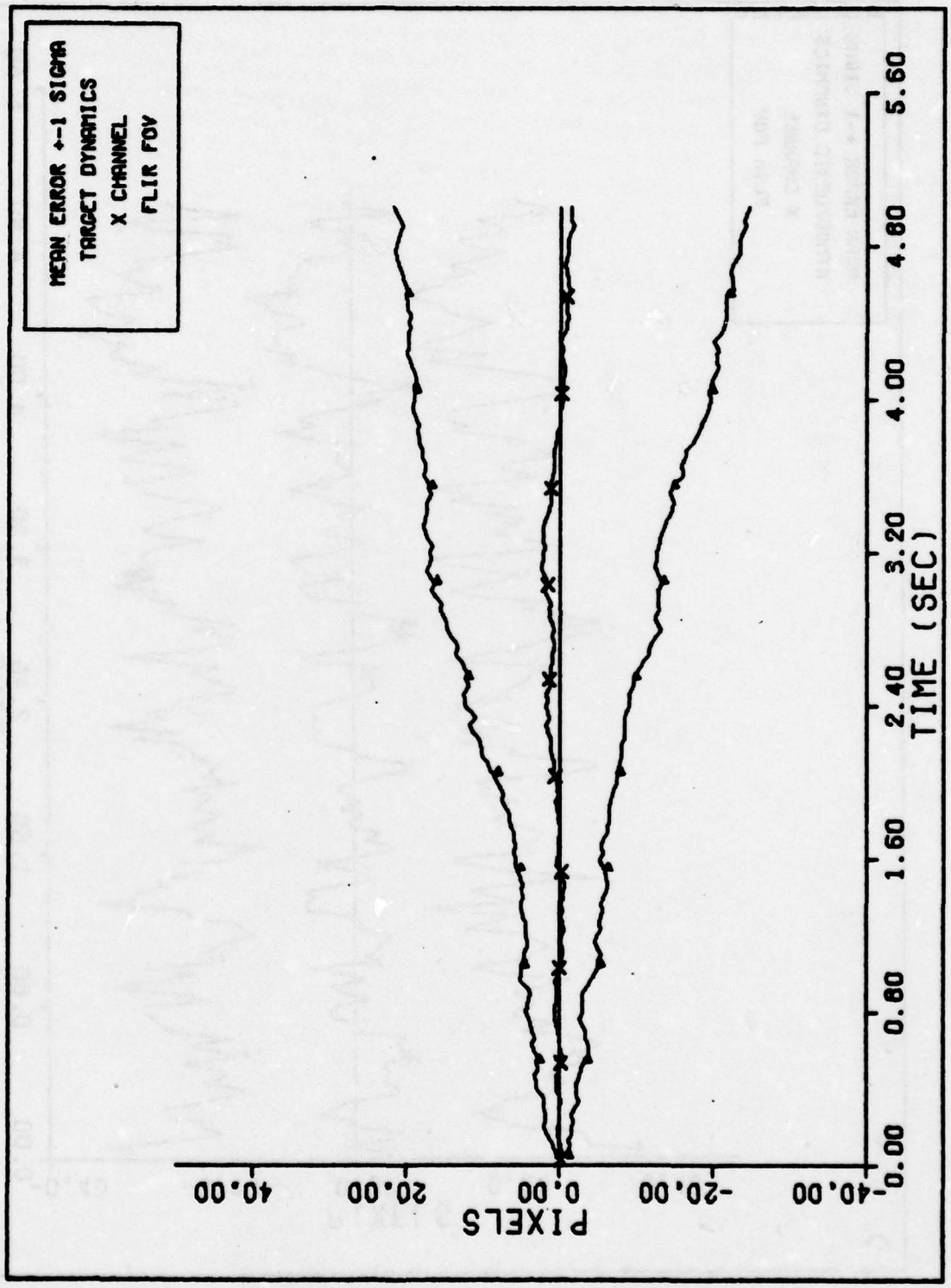
FILTER VS. ACTUAL SIGMA PLOT (S/N = /)

Figure 24d. Case 26 Performance Plot



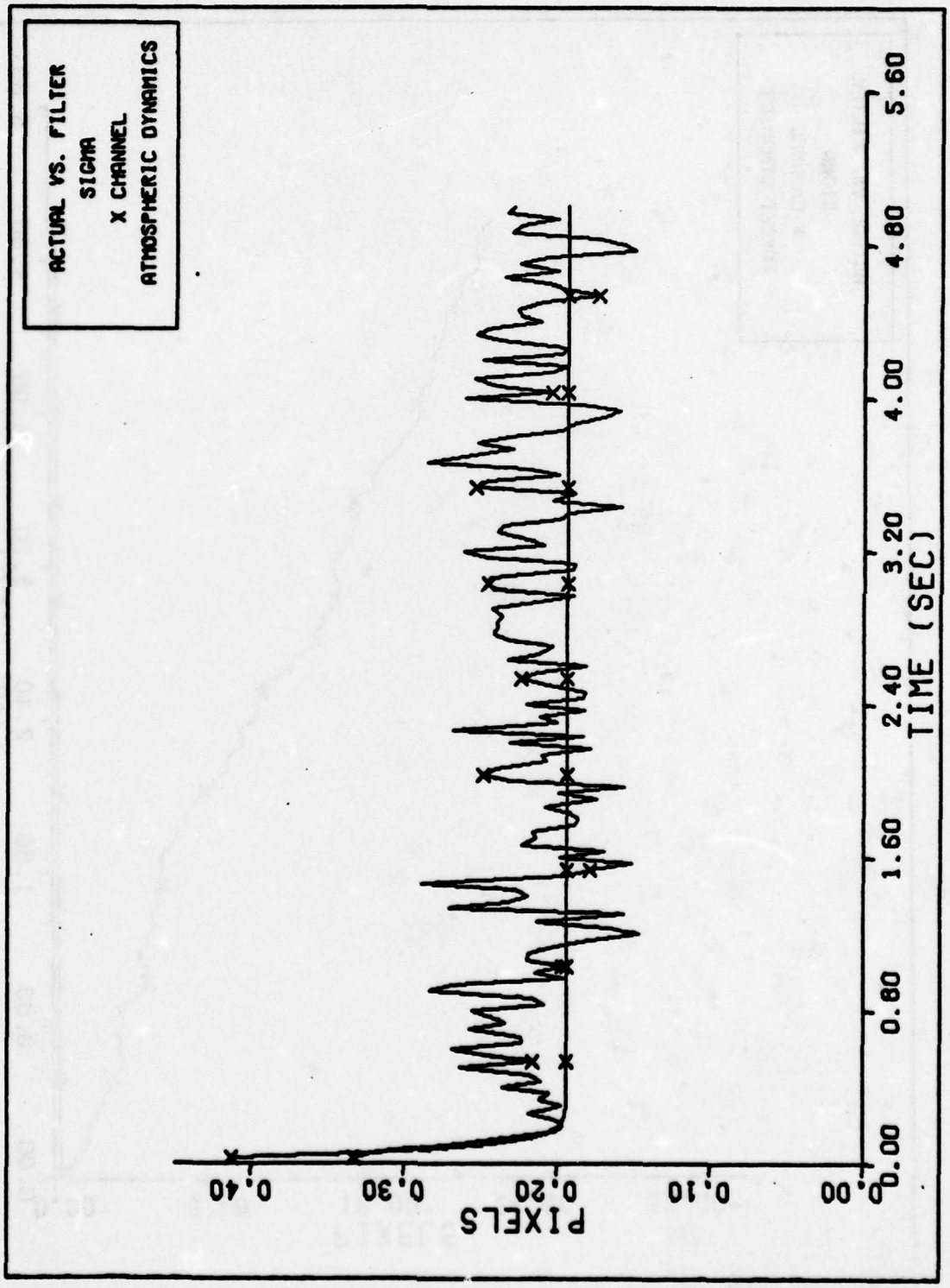
X CHANNEL ATMOSPHERICS ERROR (S/N= 2)

Figure 25a. Case 27 Performance Plot



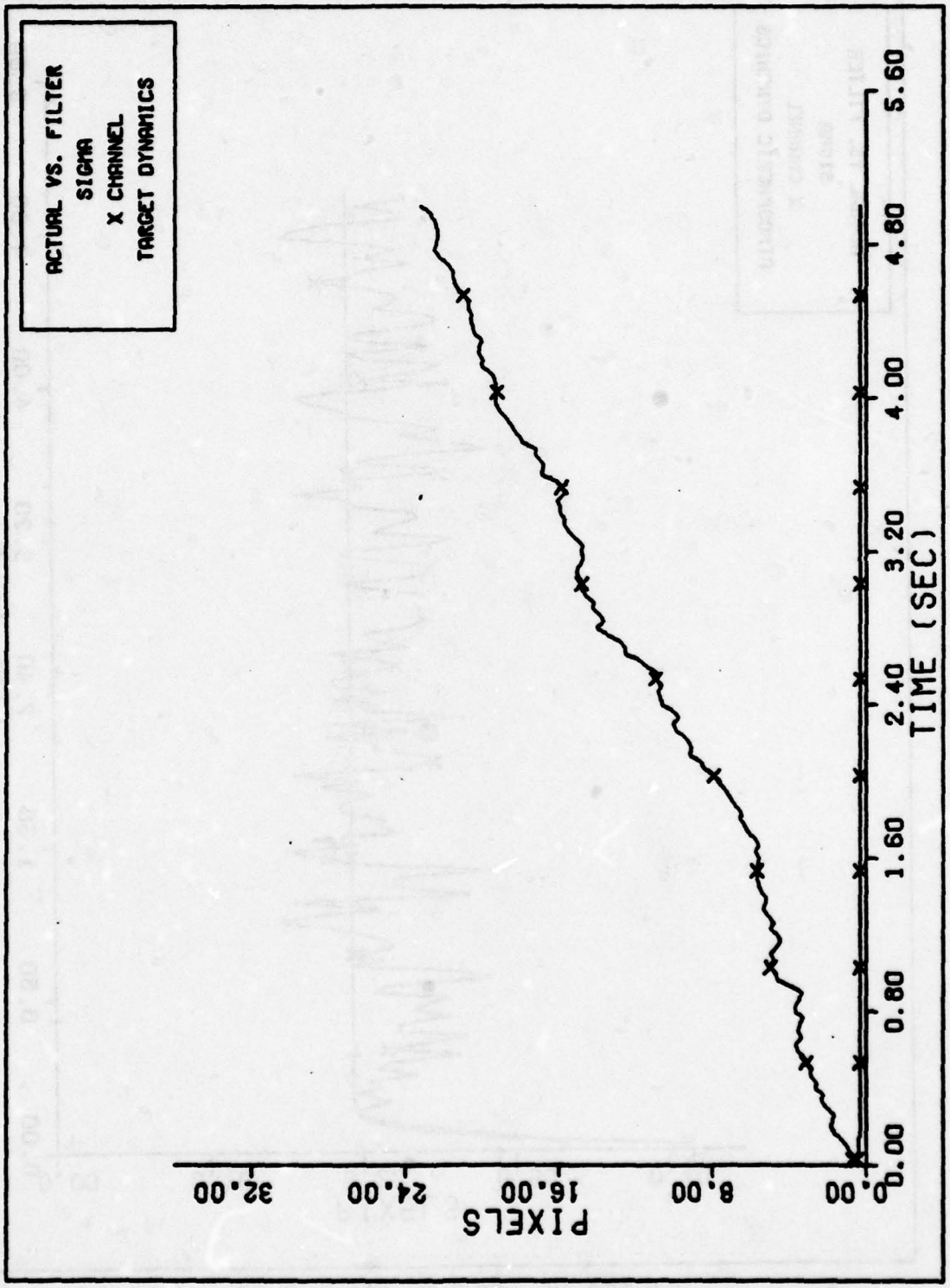
X CHANNEL DYNAMICS ERROR (S/N= 2)

Figure 25b. Case 27 Performance Plot



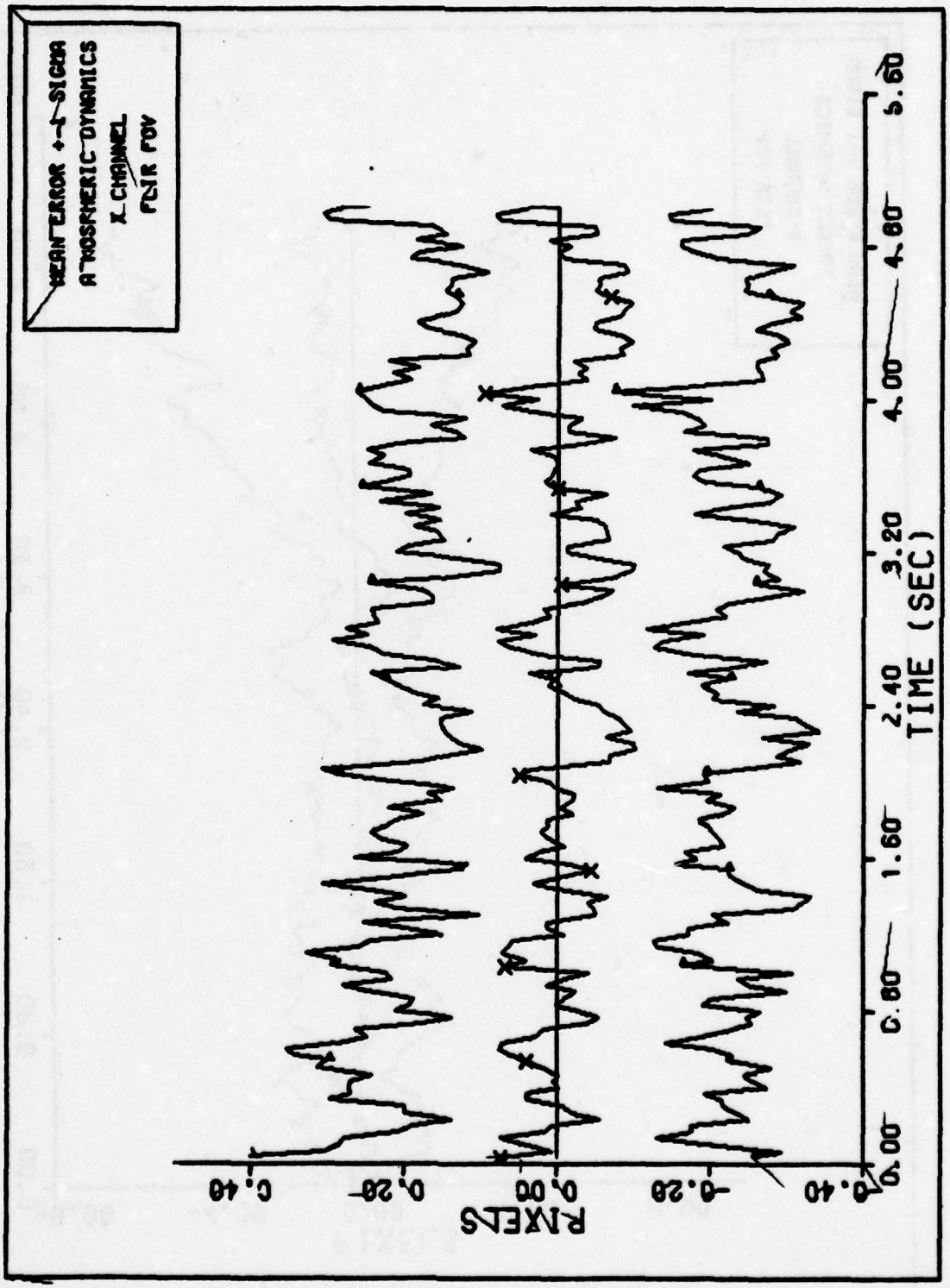
FILTER VS. ACTUAL SIGMA PLOT (S/N = 2)

Figure 25c. Case 27 Performance Plot



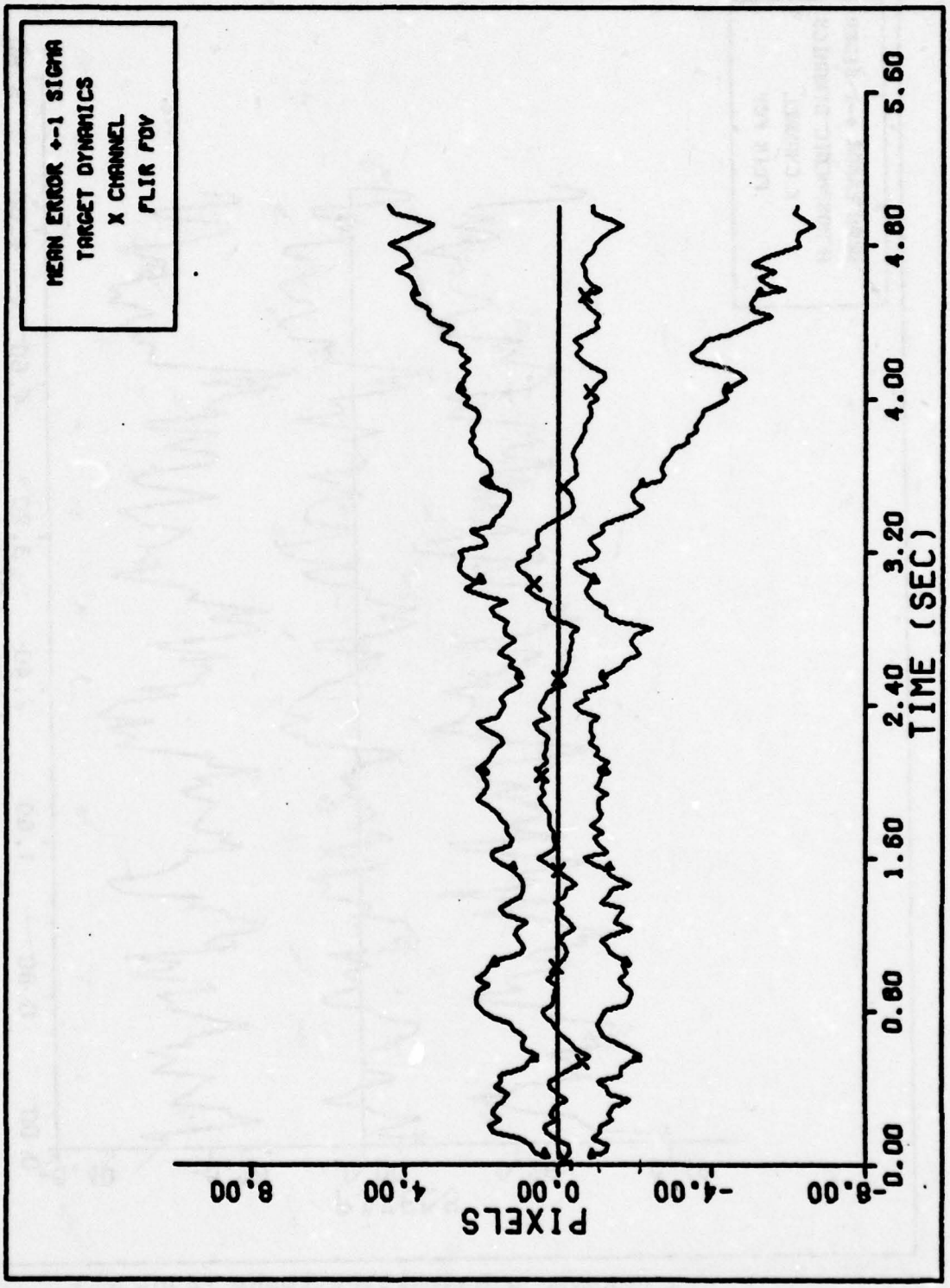
FILTER VS. ACTUAL SIGMA PLOT (S/N = 2)

Figure 25d. Case 27 Performance Plot



X CHANNEL ATMOSPHERICS ERROR S/N= 2 1

Figure 26a. Case 28 Performance Plot



X CHANNEL DYNAMICS ERROR (S/N= 2)

Figure 26b. Case 28 Performance Plot

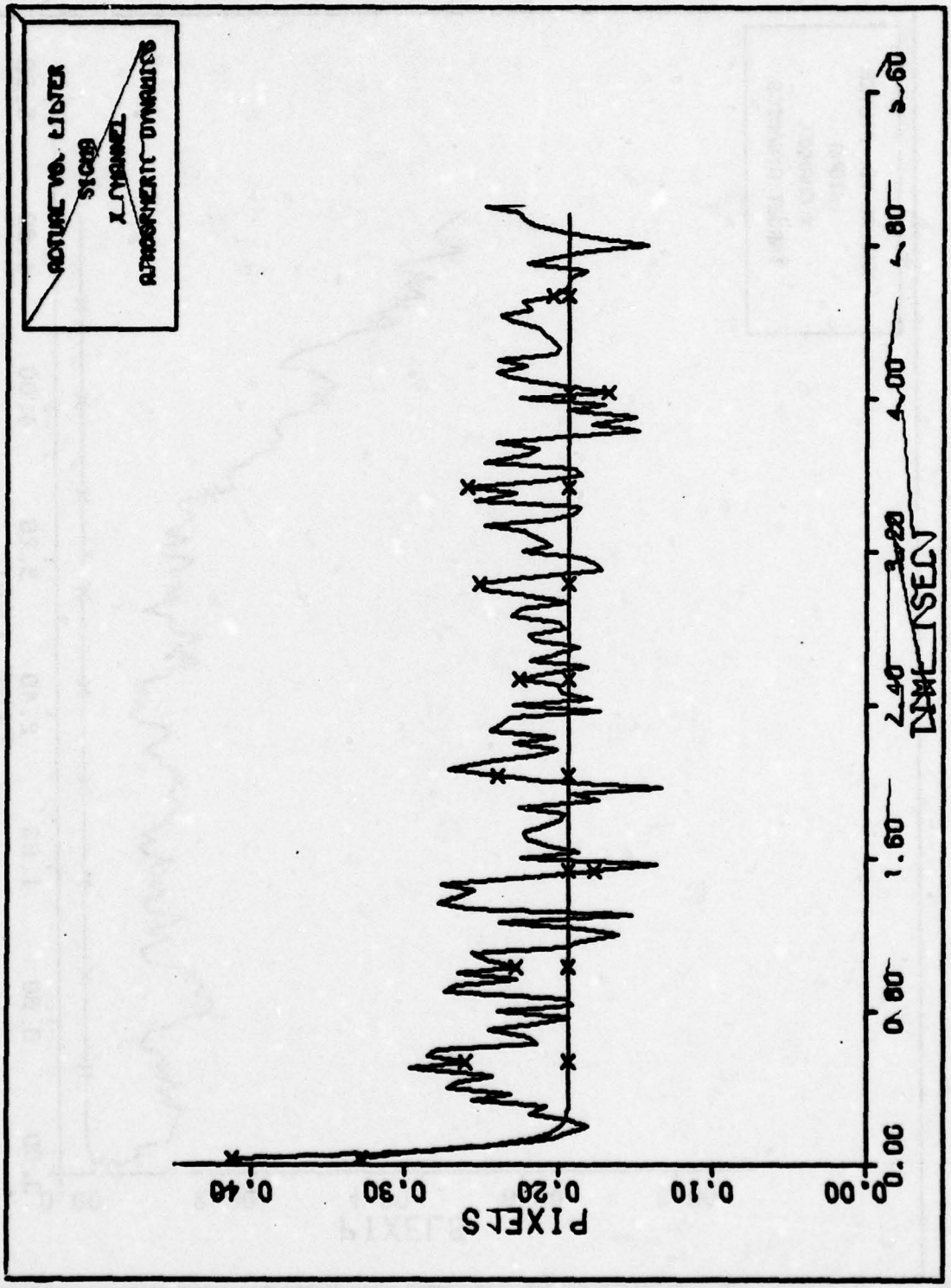
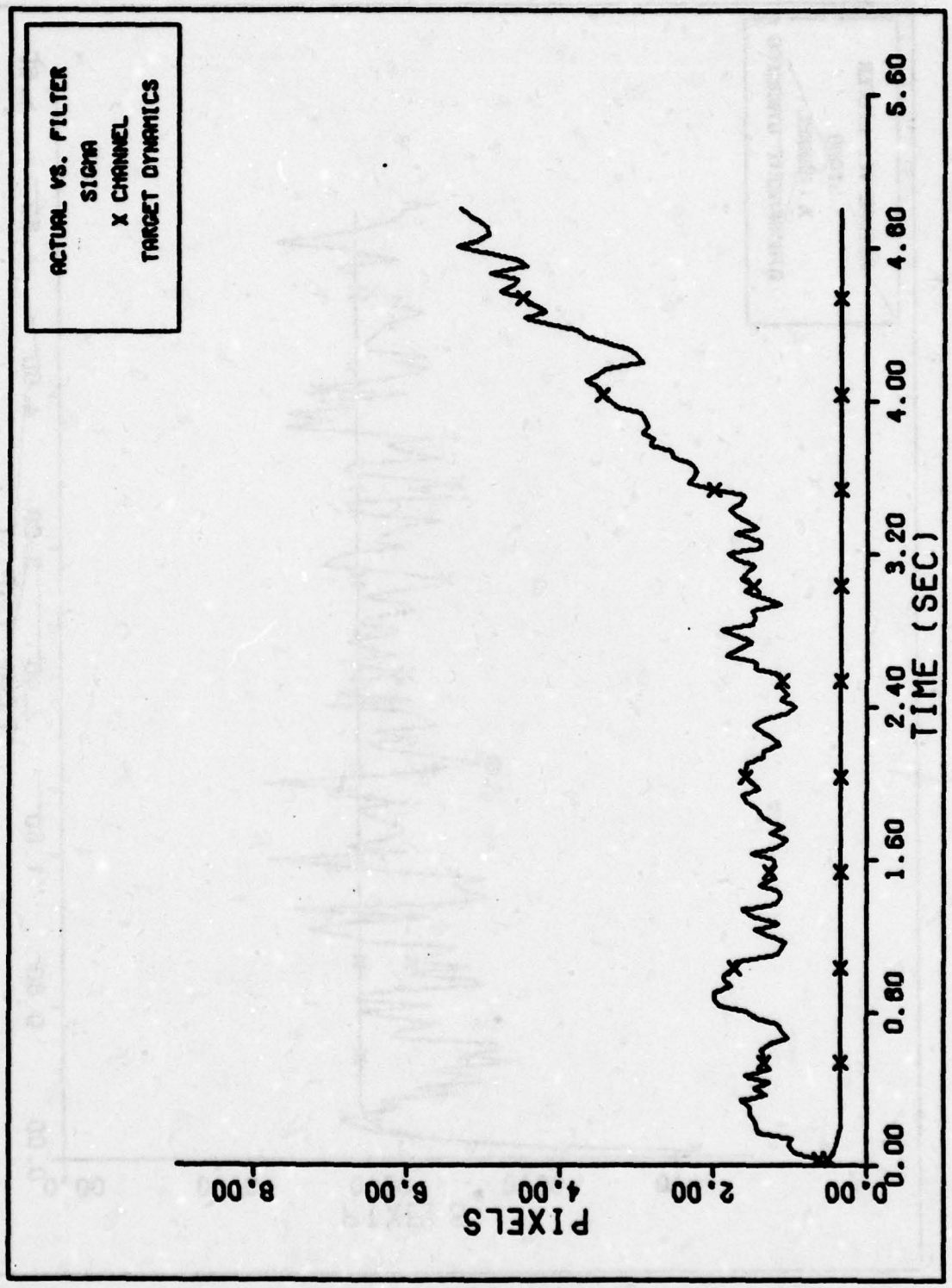
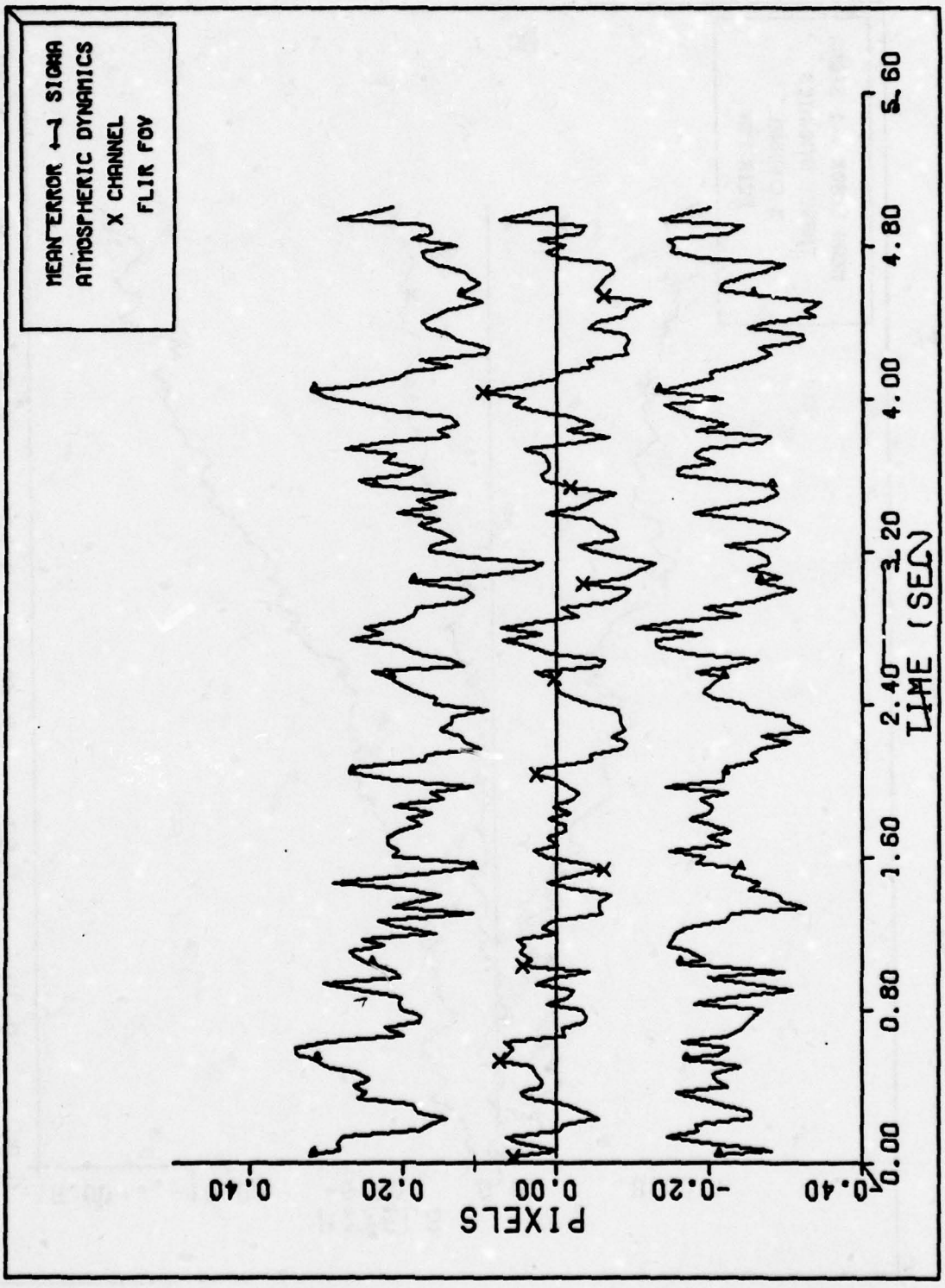


Figure 26c. Case 28 Performance Plot



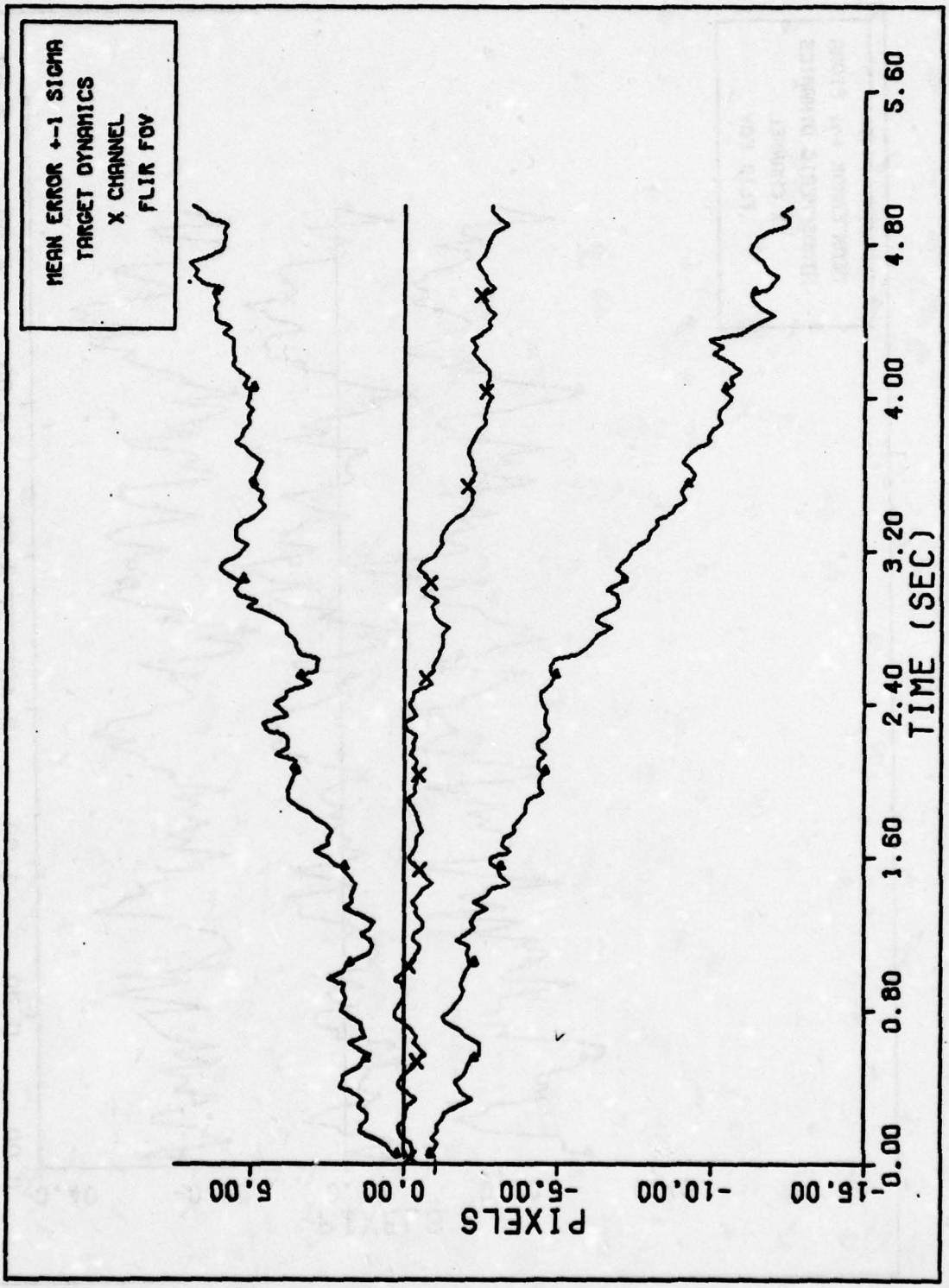
FILTER VS. ACTUAL SIGMA PLOT (S/N = 2)

Figure 26d. Case 28 Performance Plot



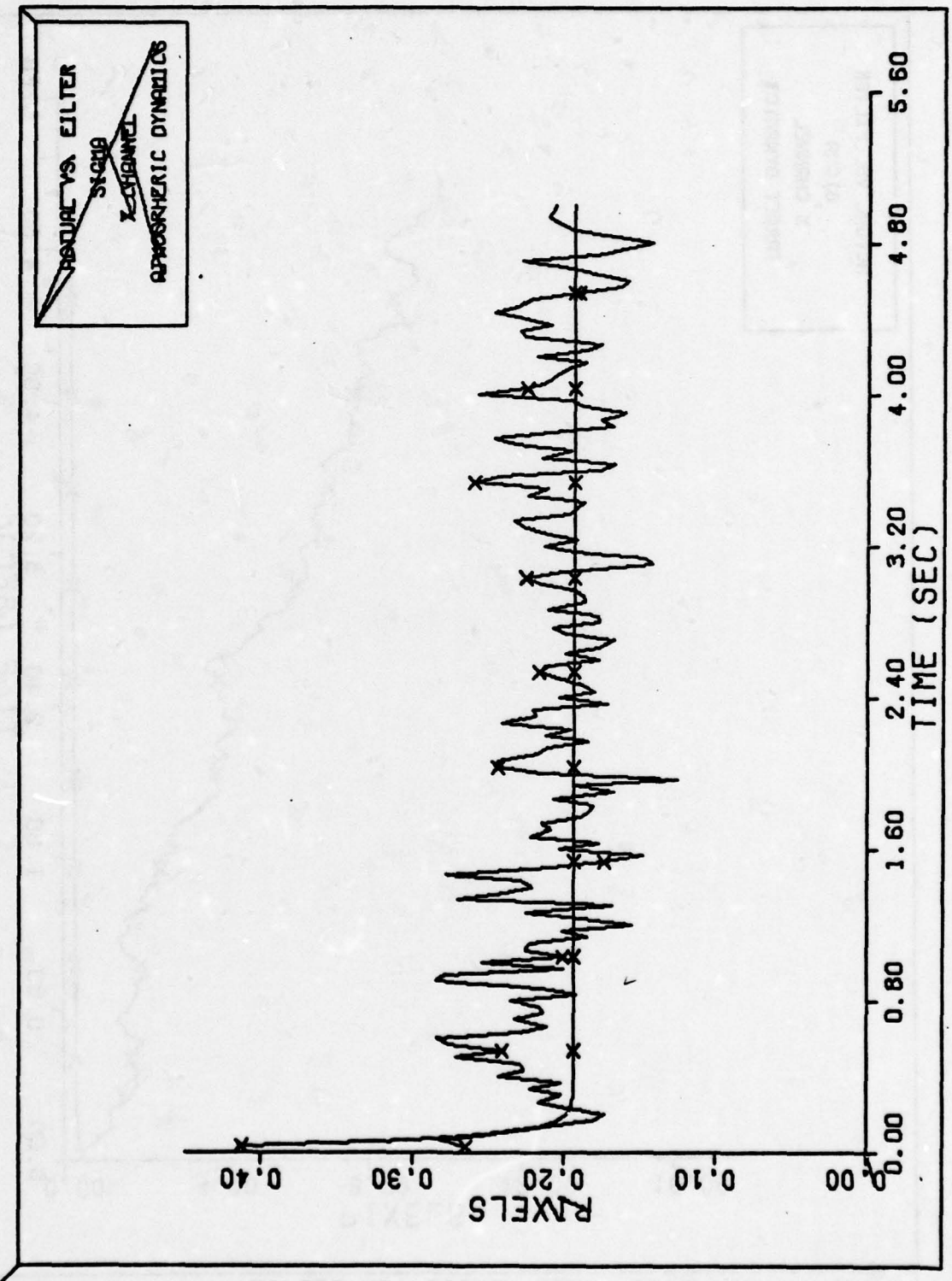
X CHANNEL ATMOSPHERICS ERROR (S/N= 2)

Figure 27a. Case 29 Performance Plot



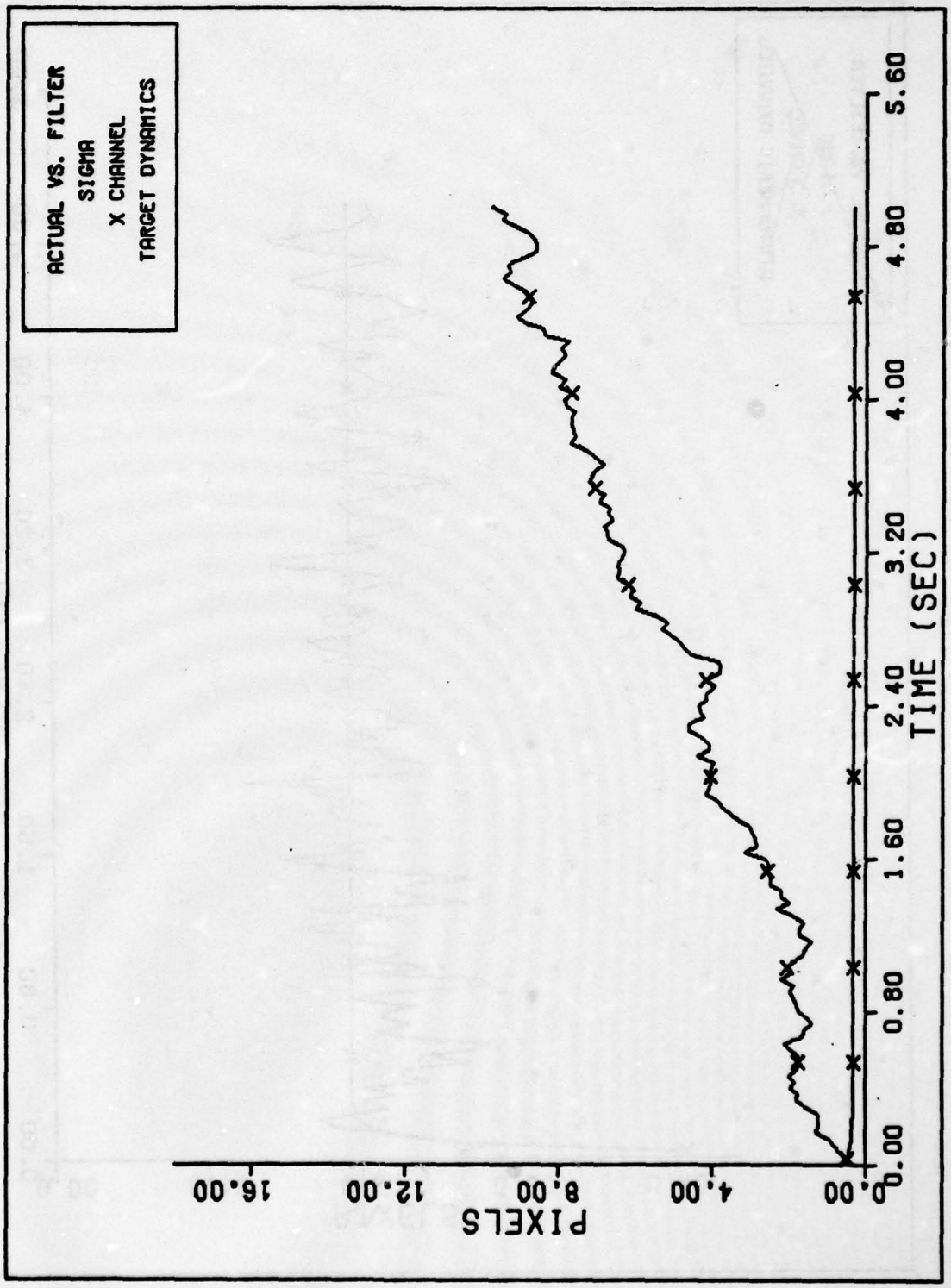
X CHANNEL DYNAMICS ERROR (S/N= 2)

Figure 27b. Case 29 Performance Plot



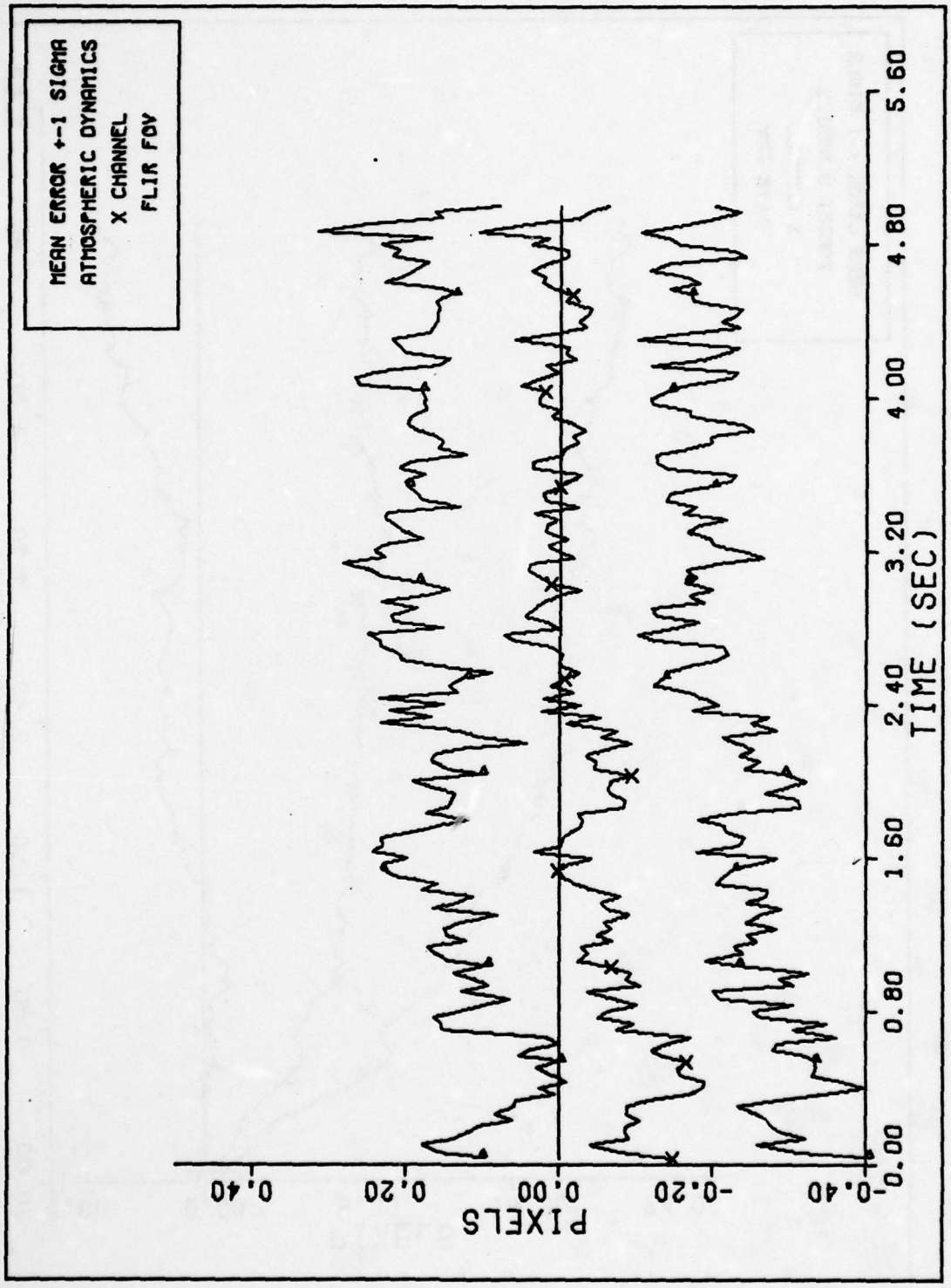
FILTER VS. ACTUAL SIGMA PLOT (S/N = 2)

Figure 27c. Case 29 Performance Plot



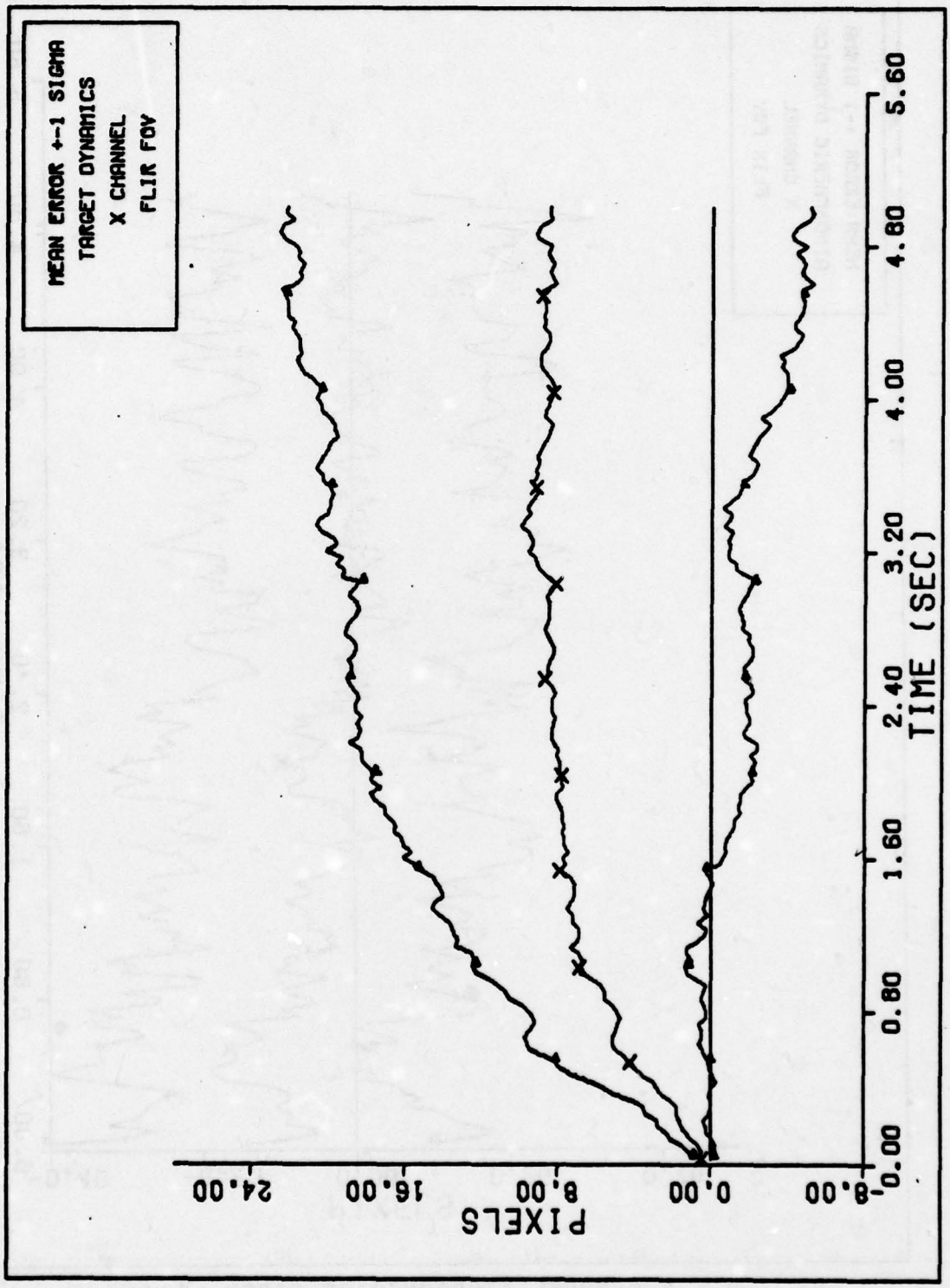
FILTER VS. ACTUAL SIGMA PLOT (S/N = 2)

Figure 27d. Case 29 Performance Plot



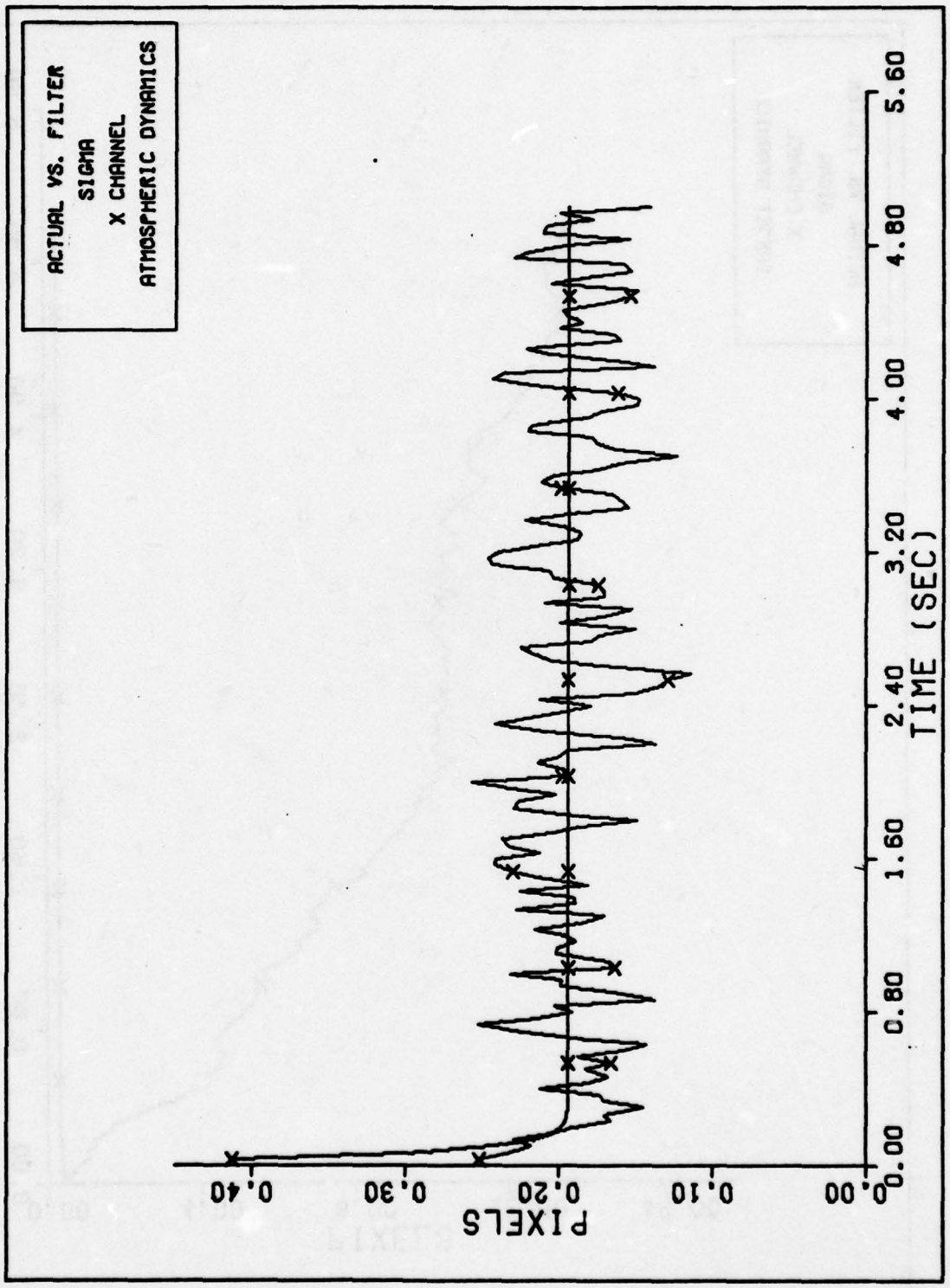
X CHANNEL ATMOSPHERICS ERROR (S/N= 2)

Figure 28a. Case 30 Performance Plot



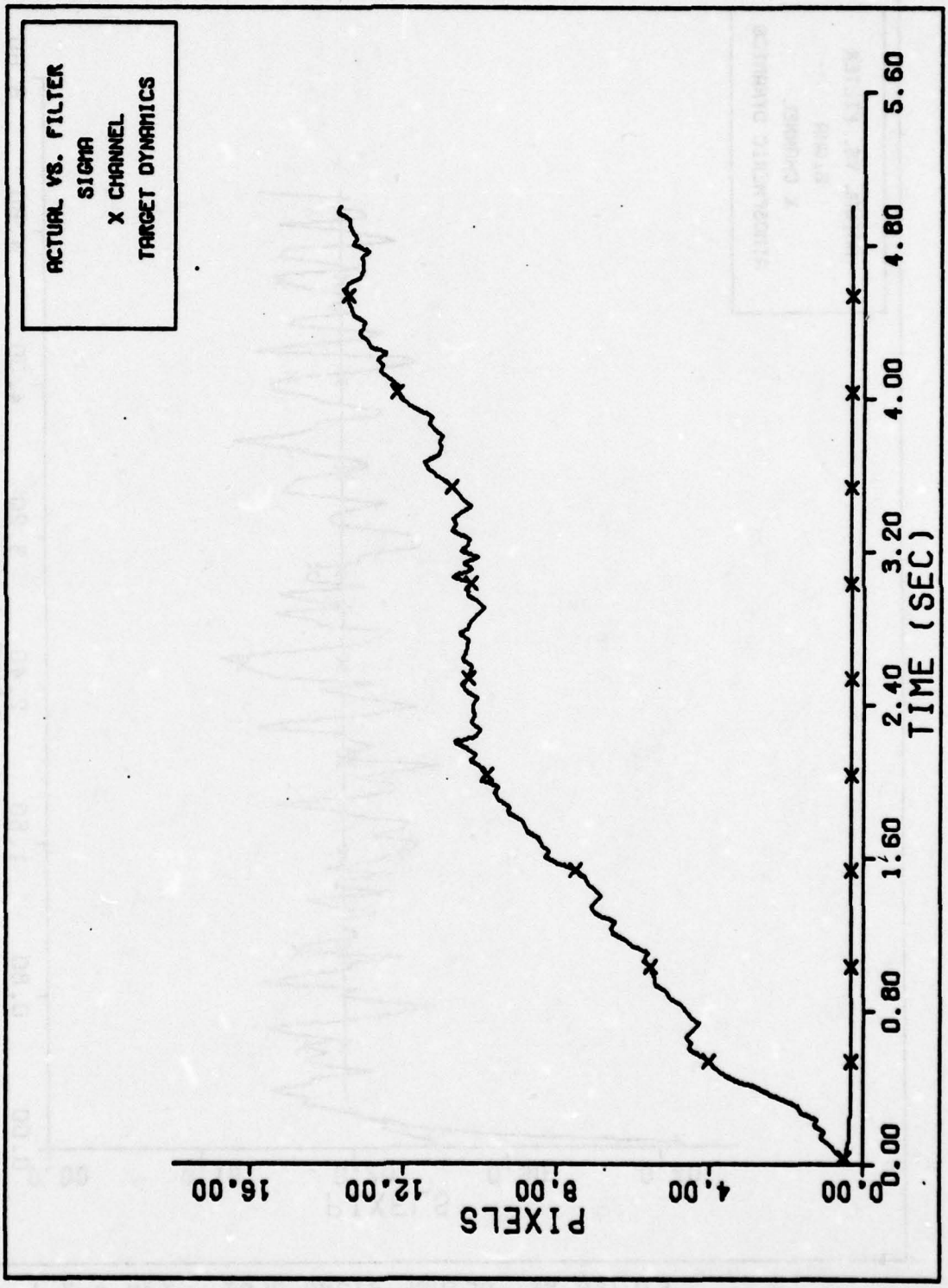
X CHANNEL DYNAMICS ERROR (S/N= 2)

Figure 28b. Case 30 Performance Plot



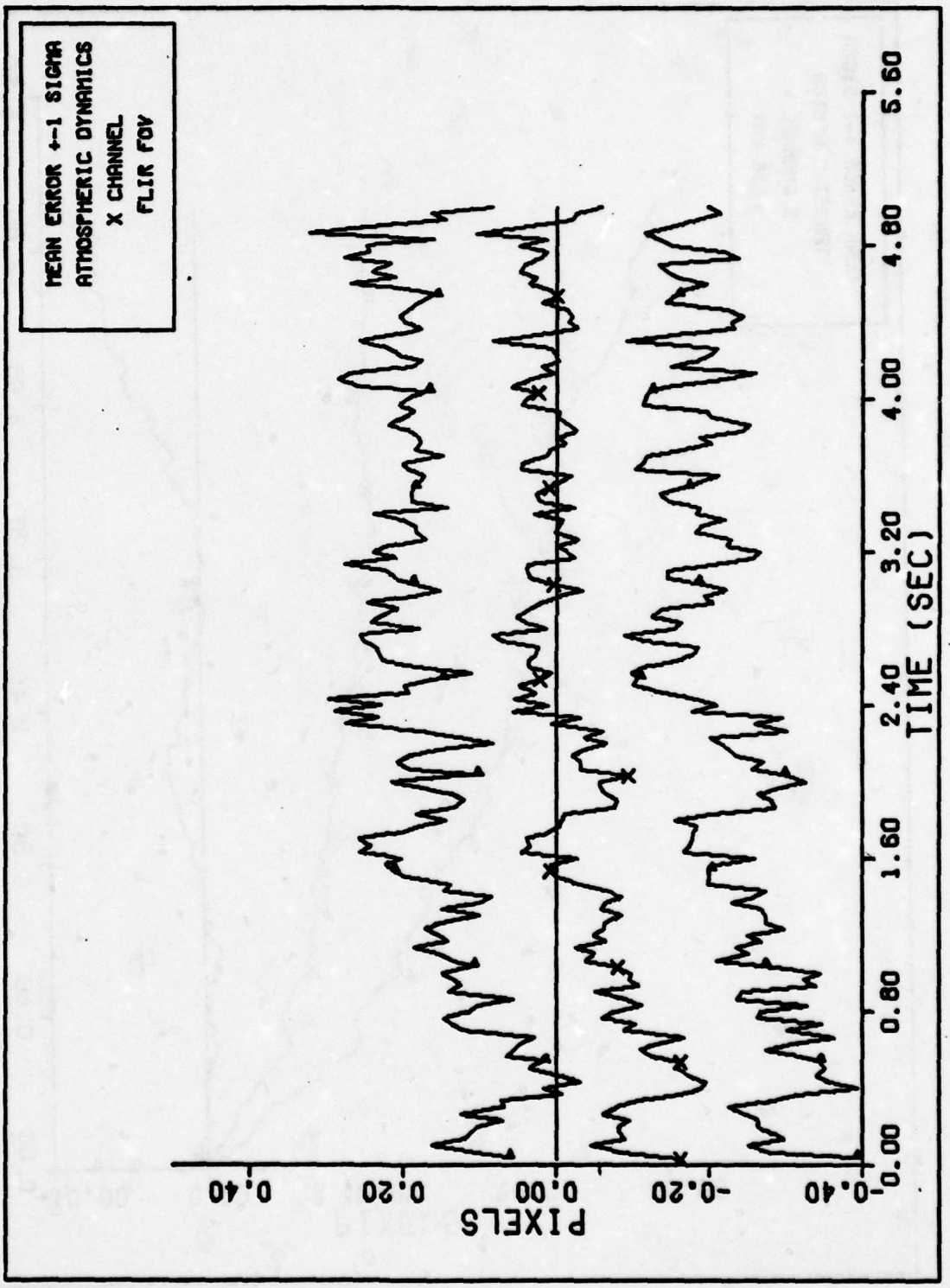
FILTER VS. ACTUAL SIGMA PLOT (S/N = 2)

Figure 28c. Case 30 Performance Plot



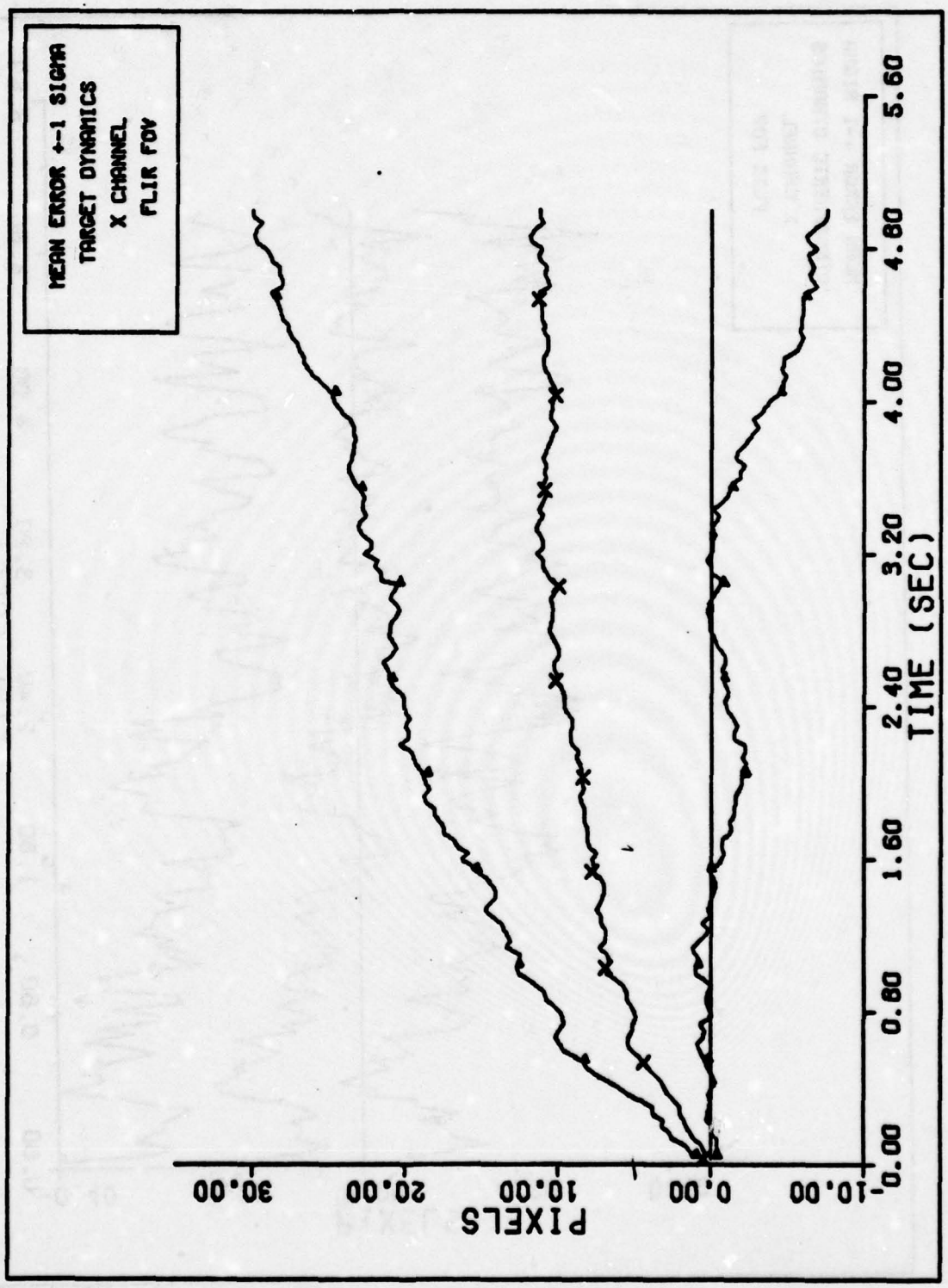
FILTER VS. ACTUAL SIGMA PLOT (S/N = 2)

Figure 28d. Case 30 Performance Plot



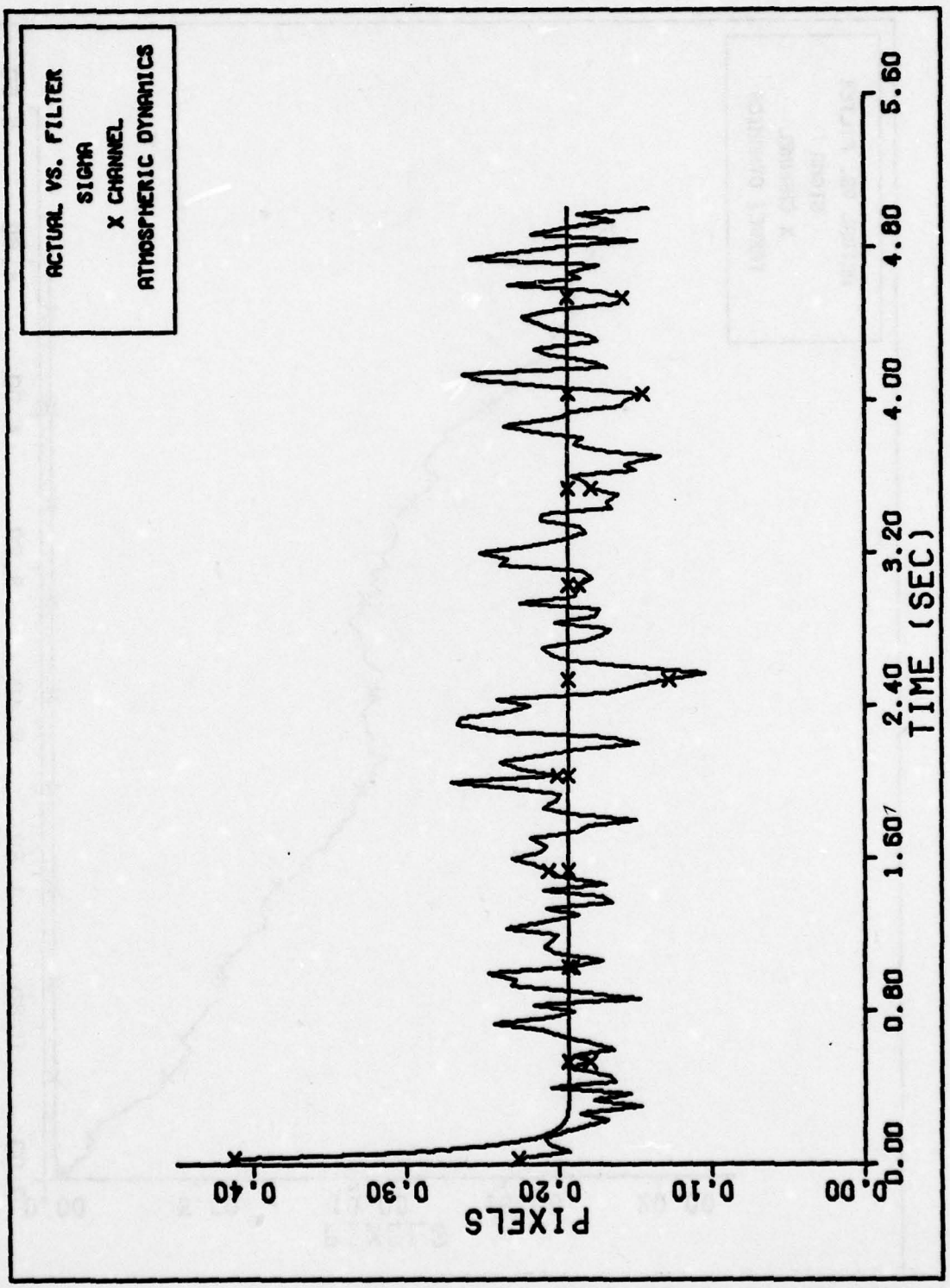
X CHANNEL ATMOSPHERICS ERROR (S/N= 2)

Figure 29a. Case 31 Performance Plot



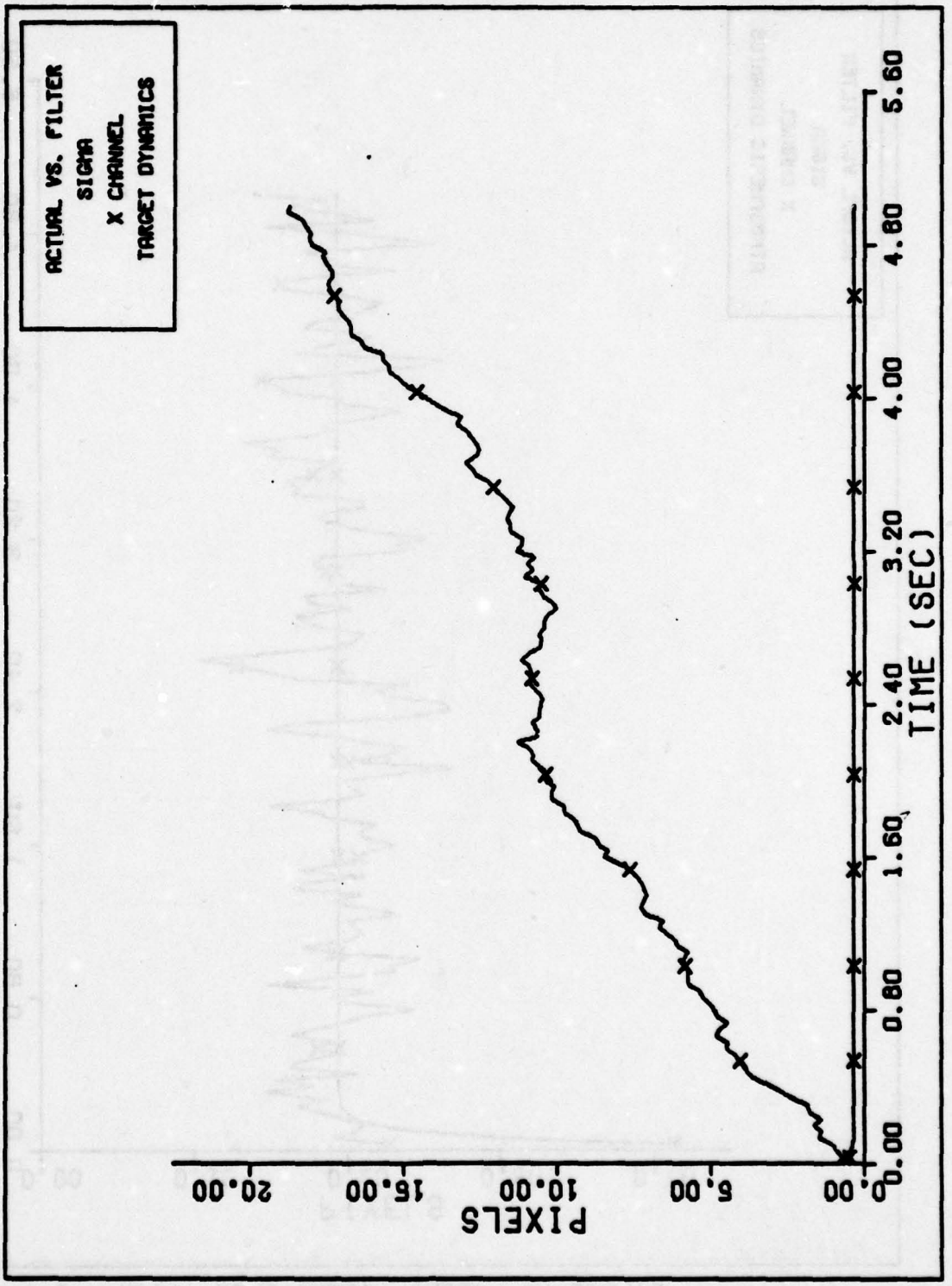
X CHANNEL DYNAMICS ERROR (S/N= 2)

Figure 29b. Case 31 Performance Plot



FILTER VS. ACTUAL SIGMA PLOT (S/N = 2)

Figure 29c. Case 31 Performance Plot



FILTER VS. ACTUAL SIGMA PLOT (S/N = 2)

Figure 29d. Case 31 Performance Plot

Appendix K

Input Parameters for Cases 32-37

Symbol	Case / Fortran Code	Refer to List of Symbols for Description of Variables					
		8 32	6 33	6 34	6 35	8 36	6 37
σ_D	SIGS1	0	0	0	0	0	0
σ_N	SIGMAB	2	2	2	2	2	2
σ_f	SIGFLR	0	0	0	0	0	0
I_{max}	IMAX	25	25	25	25	25	25
σ_A	SIGAT	.2	.2	.2	.2	.2	.2
-	NRUN	20	20	20	20	20	20
t_f	TFINAL	5	5	5	5	5	5
σ_{gT}	SIGMS	1	1	1	1	1	1
AR_T	ASPRO	5	5	5	5	5	5
x_o	XO	1500	1500	20000	20000	1500	1500
y_o	YO	500	500	0	0	500	500
z_o	ZO	10000	10000	0	0	10000	10000
$\dot{x}_o$	XDOTO	-508	-500	0	0	-500	-500
$\dot{y}_o$	YDOTO	-308	-300	0	0	-300	-300
$\dot{z}_o$	ZDOTO	8	0	1000	1000	0	0
-	ISPTL	0	0	0	0	0	0
$r_{k,k+1}$	C(1)	-	-	-	-	-	-
$r_{k,k+9}$	C(2)	-	-	-	-	-	-
$r_{k,k+2}$	C(3)	-	-	-	-	-	-
$r_{k,k+10}$	C(4)	-	-	-	-	-	-
$r_{k,k+18}$	C(5)	-	-	-	-	-	-

Symbol	Fortran Code / Case	Refer to List of Symbols for Description of Variables					
		8 32	6 33	6 34	6 35	8 36	6 37
$\sigma_{\dot{x}o}$	SIGMFO	-3.	-3	-3	-3	-3	-3
AR_F	ARO	1	1	1	1	1	1
σ_{AF}	SIGF2	.2	.2	.2	.2	.2	.2
σ_v	RF	2	2	2	2	2	2
σ_{DF}	SIGF1O	600	-600	-600	-600	600	600
I_{maxF}	FIMAXO	-25	-25	-25	-25	-25	-25

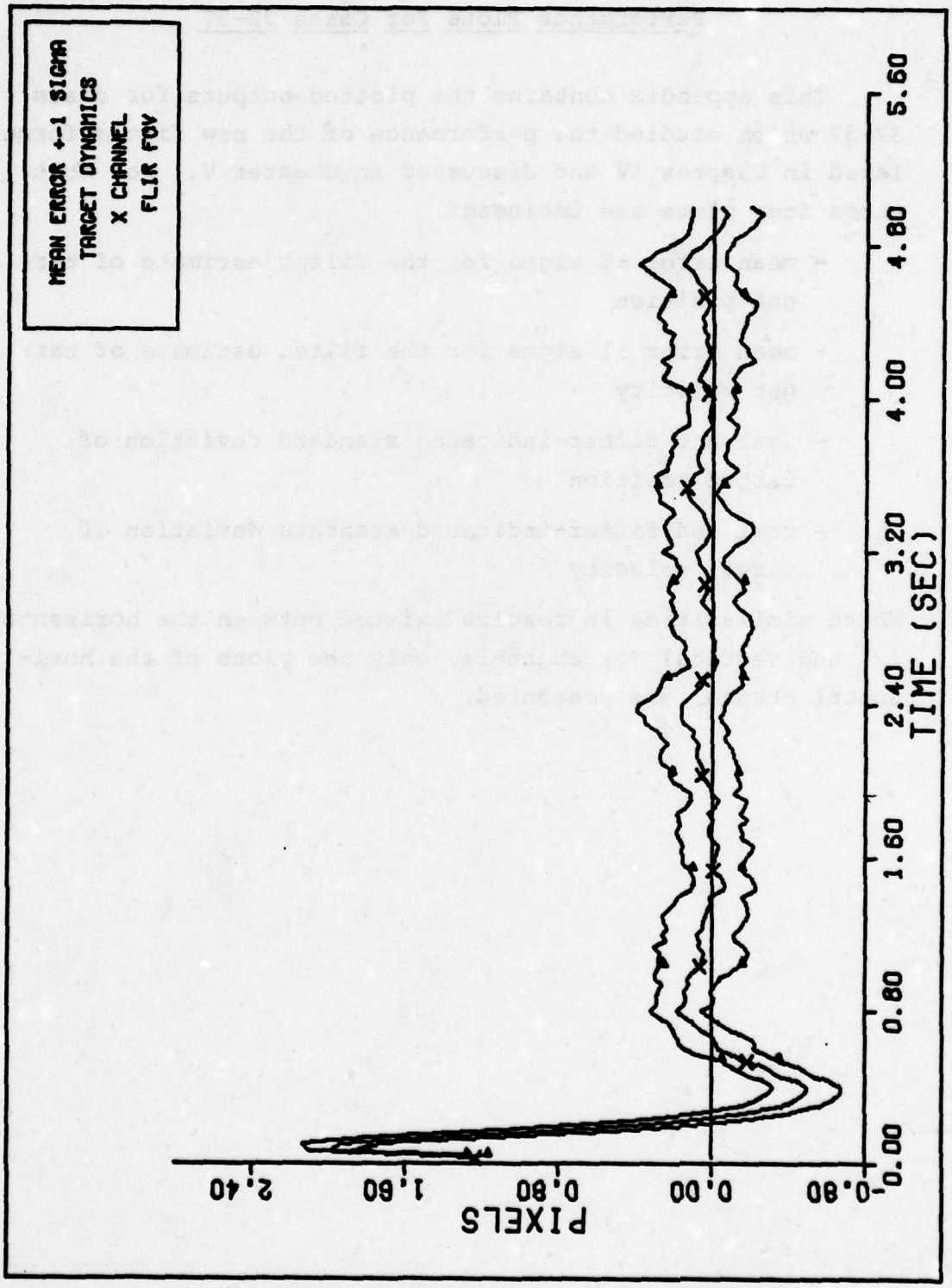
Appendix L

Performance Plots for Cases 32-37

This appendix contains the plotted outputs for cases 32-37 which studied the performance of the new filter formulated in Chapter IV and discussed in Chapter V. For most cases four plots are included:

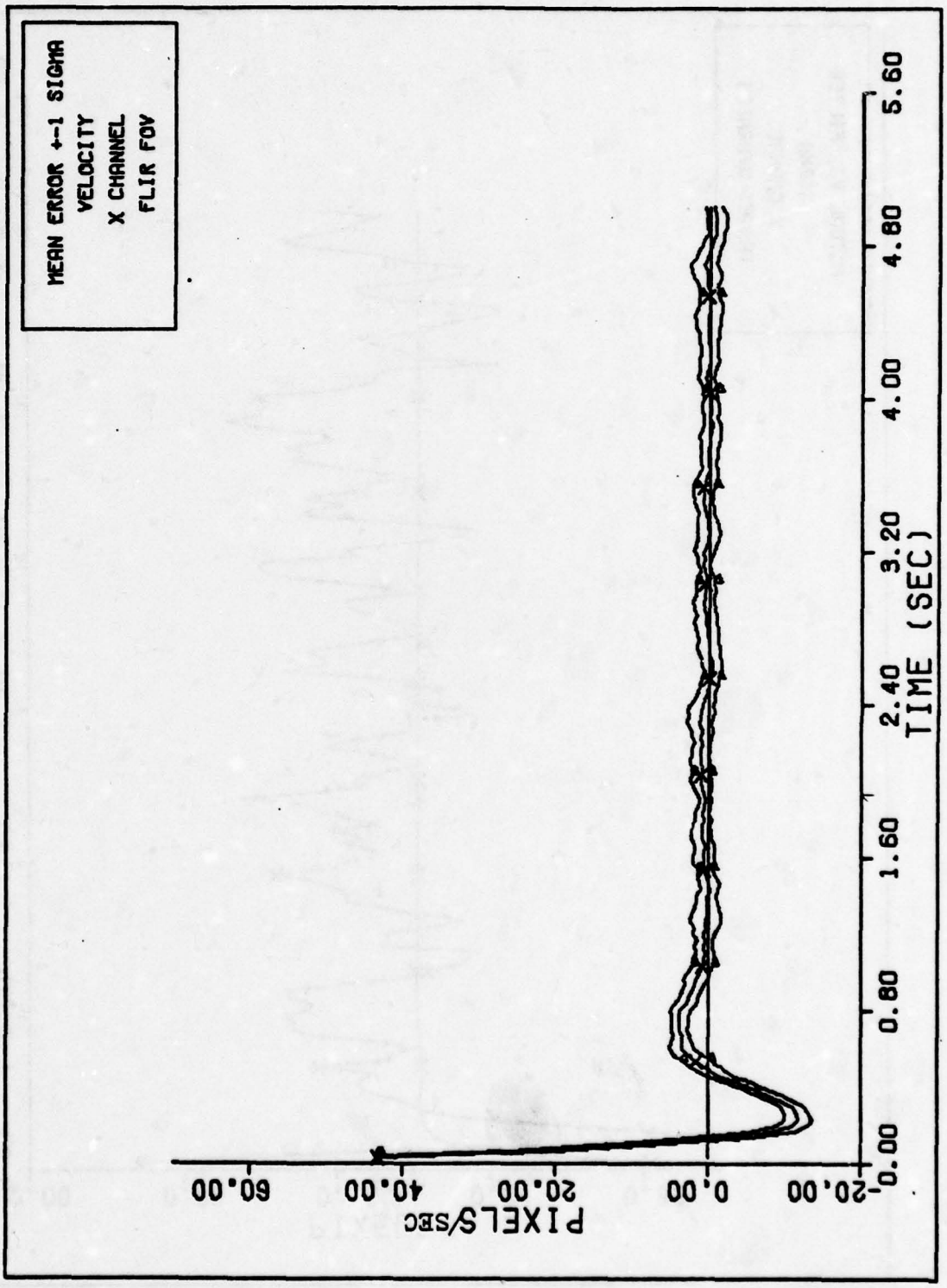
- mean error ±1 sigma for the filter estimate of target position
- mean error ±1 sigma for the filter estimate of target velocity
- real and filter-indicated standard deviation of target position
- real and filter-indicated standard deviation of target velocity

Where similarities in results existed between the horizontal (x) and vertical (y) channels, only the plots of the horizontal channel are presented.



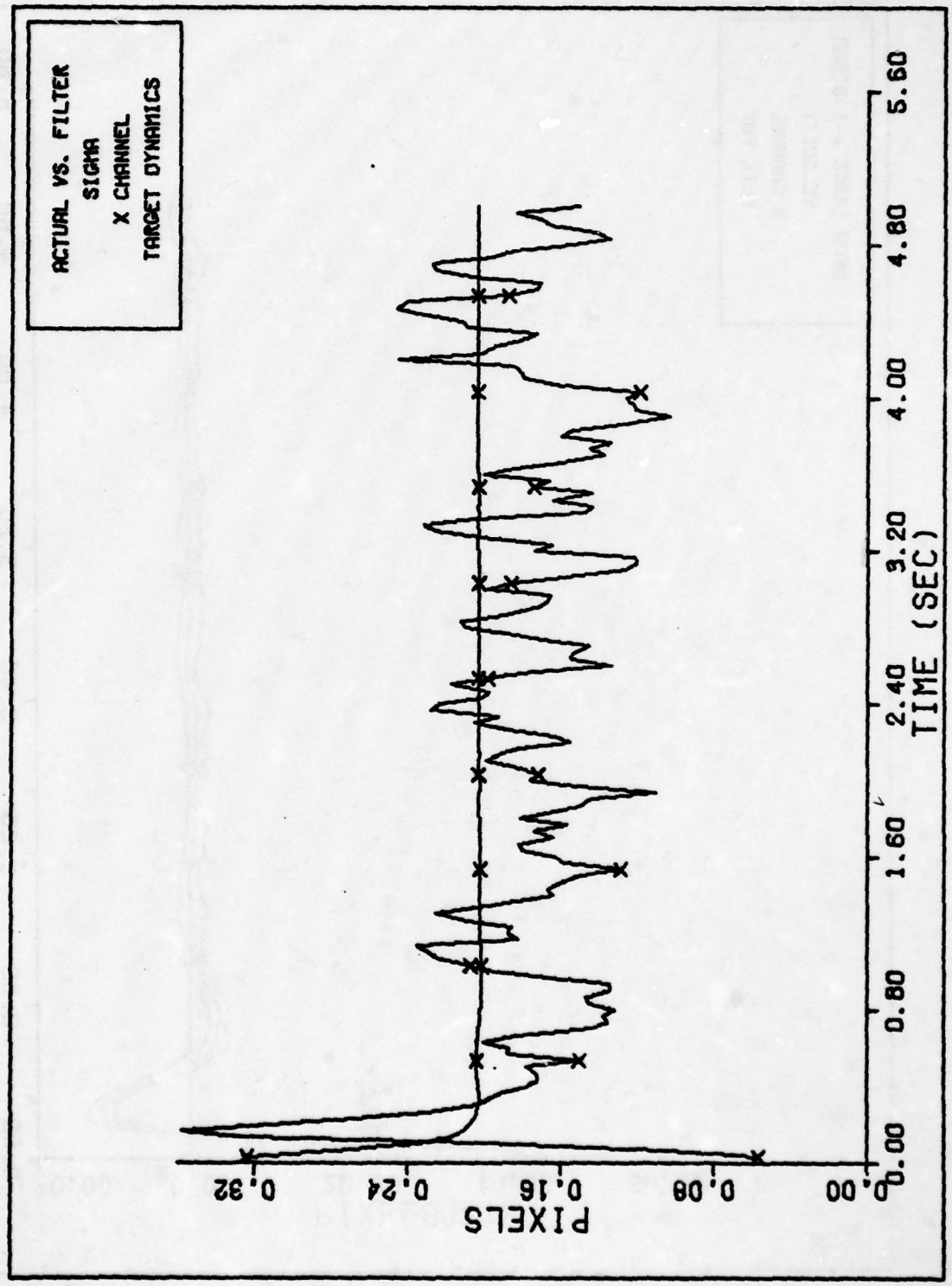
X CHANNEL DYNAMICS ERROR (S/N=12.5)

Figure 30a. Case 32 Performance Plot



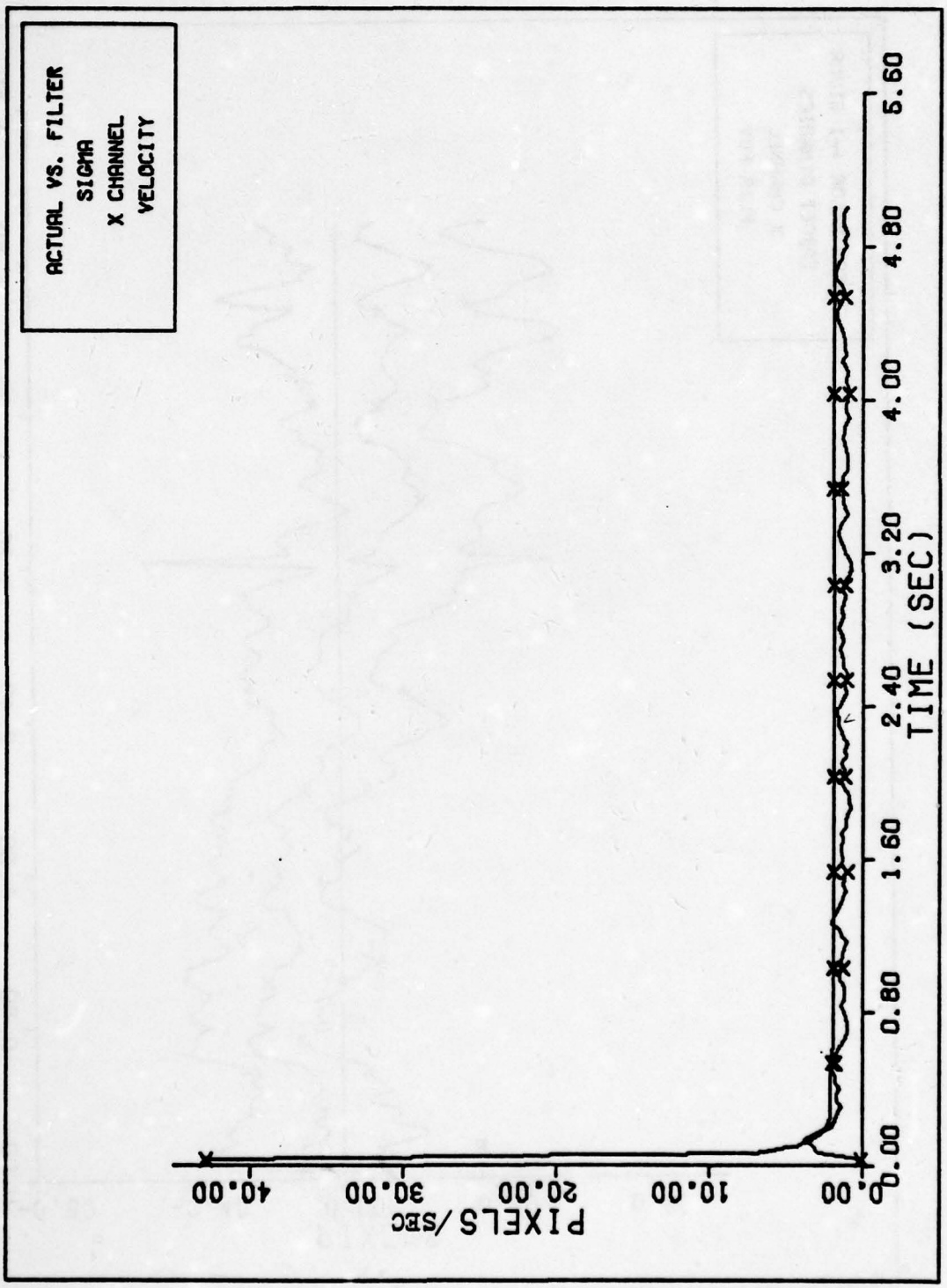
X CHANNEL VELOCITY ERROR

Figure 30b. Case 32 Performance Plot



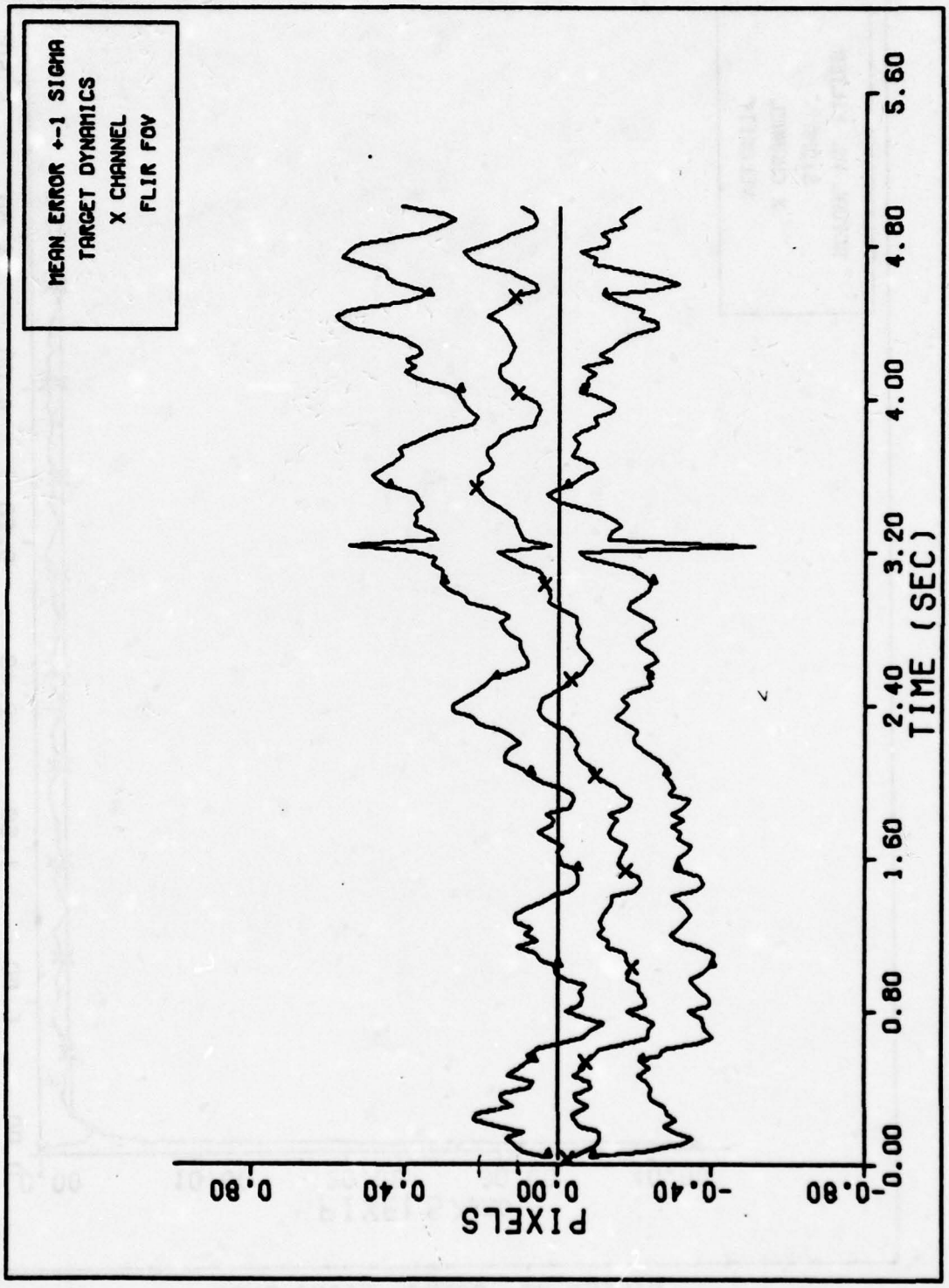
FILTER VS. ACTUAL SIGMA PLOT

Figure 30c. Case 32 Performance Plot



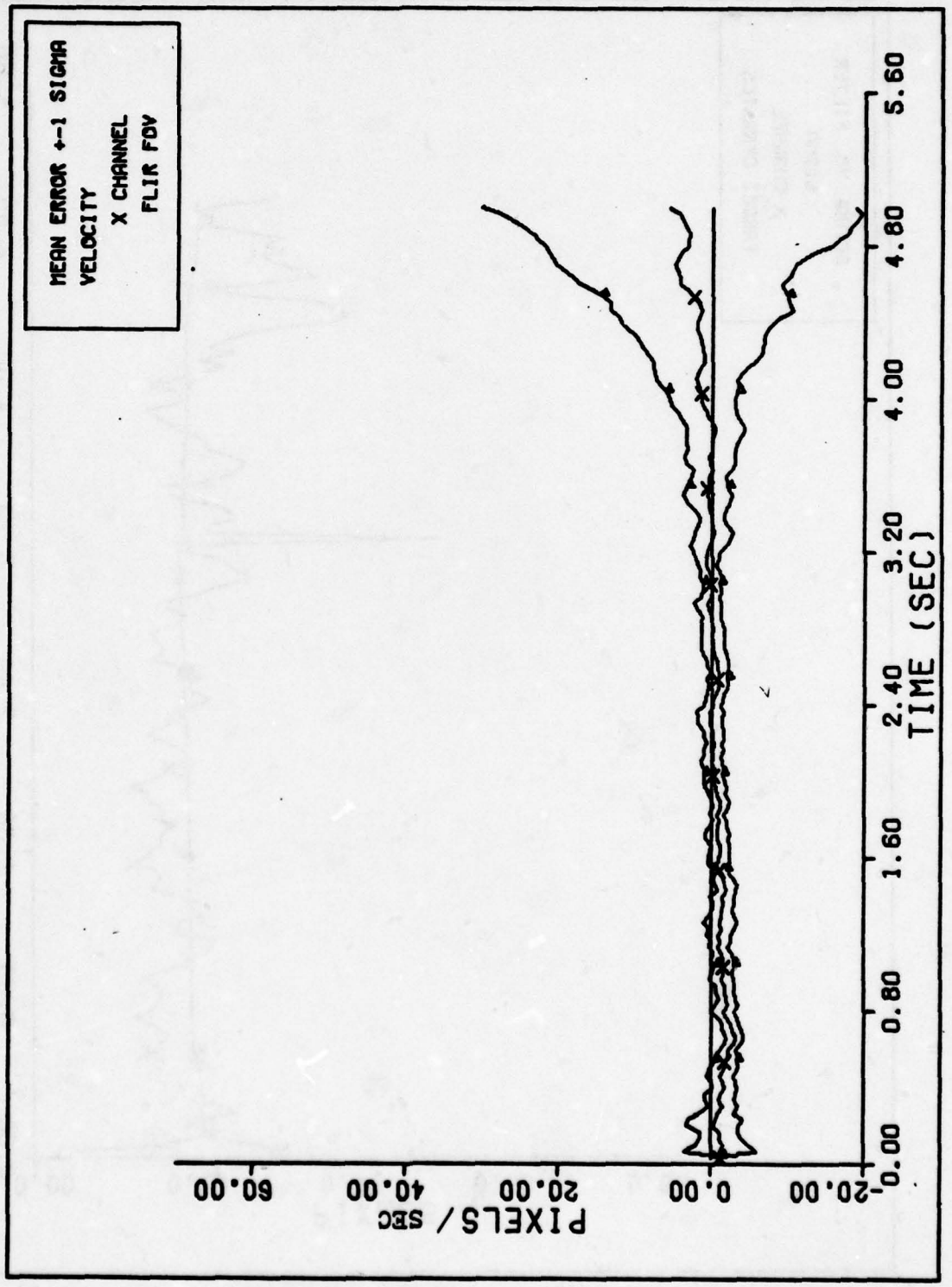
FILTER VS. ACTUAL SIGMA PLOT

Figure 30d. Case 32 Performance Plot



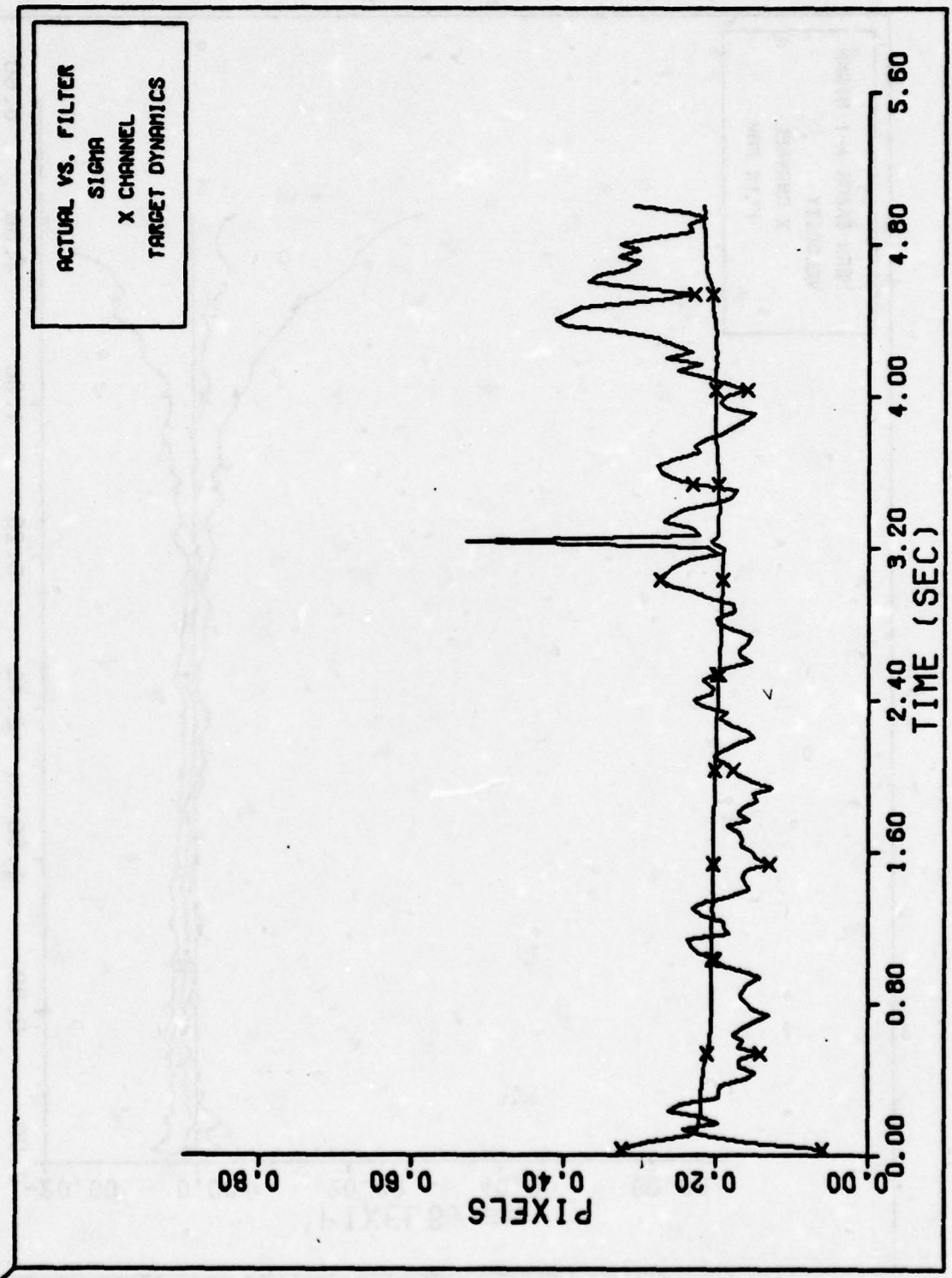
X CHANNEL DYNAMICS ERROR (S/N=12.5)

Figure 31a. Case 33 Performance Plot



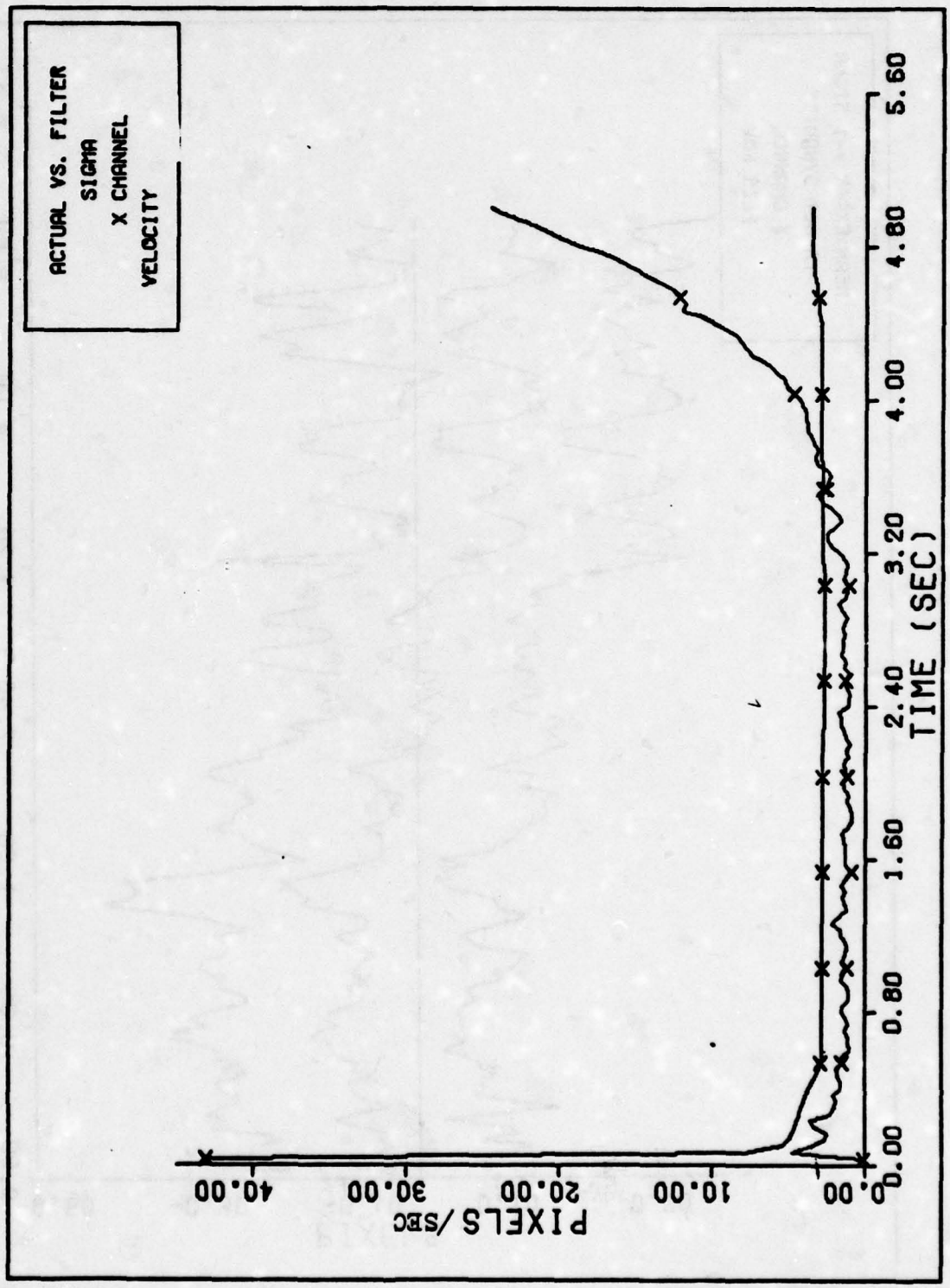
Y CHANNEL VELOCITY ERROR

Figure 31b. Case 33 Performance Plot



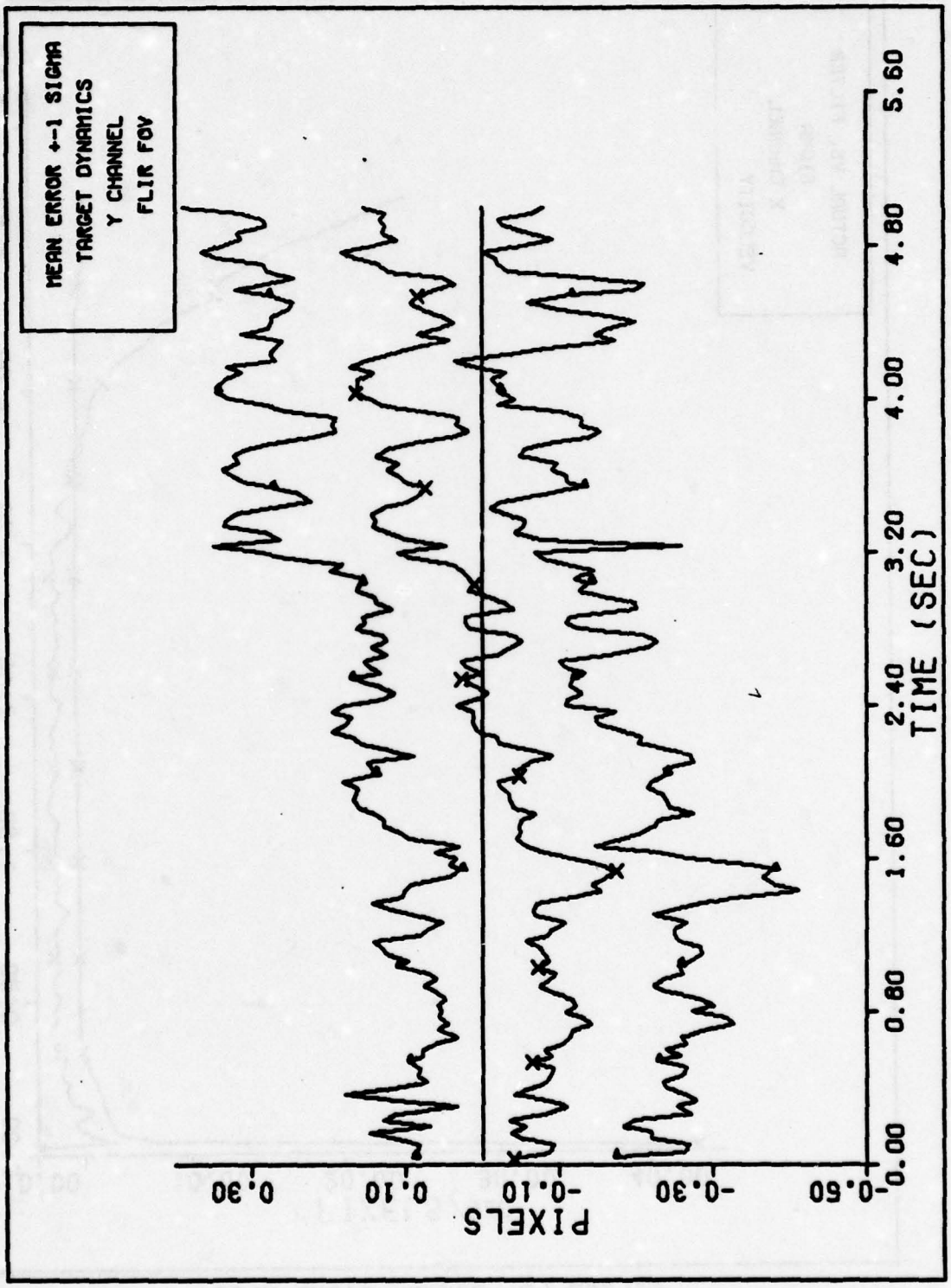
FILTER VS. ACTUAL SIGMA PLOT

Figure 31c. Case 33 Performance Plot



FILTER VS. ACTUAL SIGMA PLOT

Figure 31d. Case 33 Performance Plot



Y CHANNEL DYNAMICS ERROR

Figure 31e. Case 33 Performance Plot

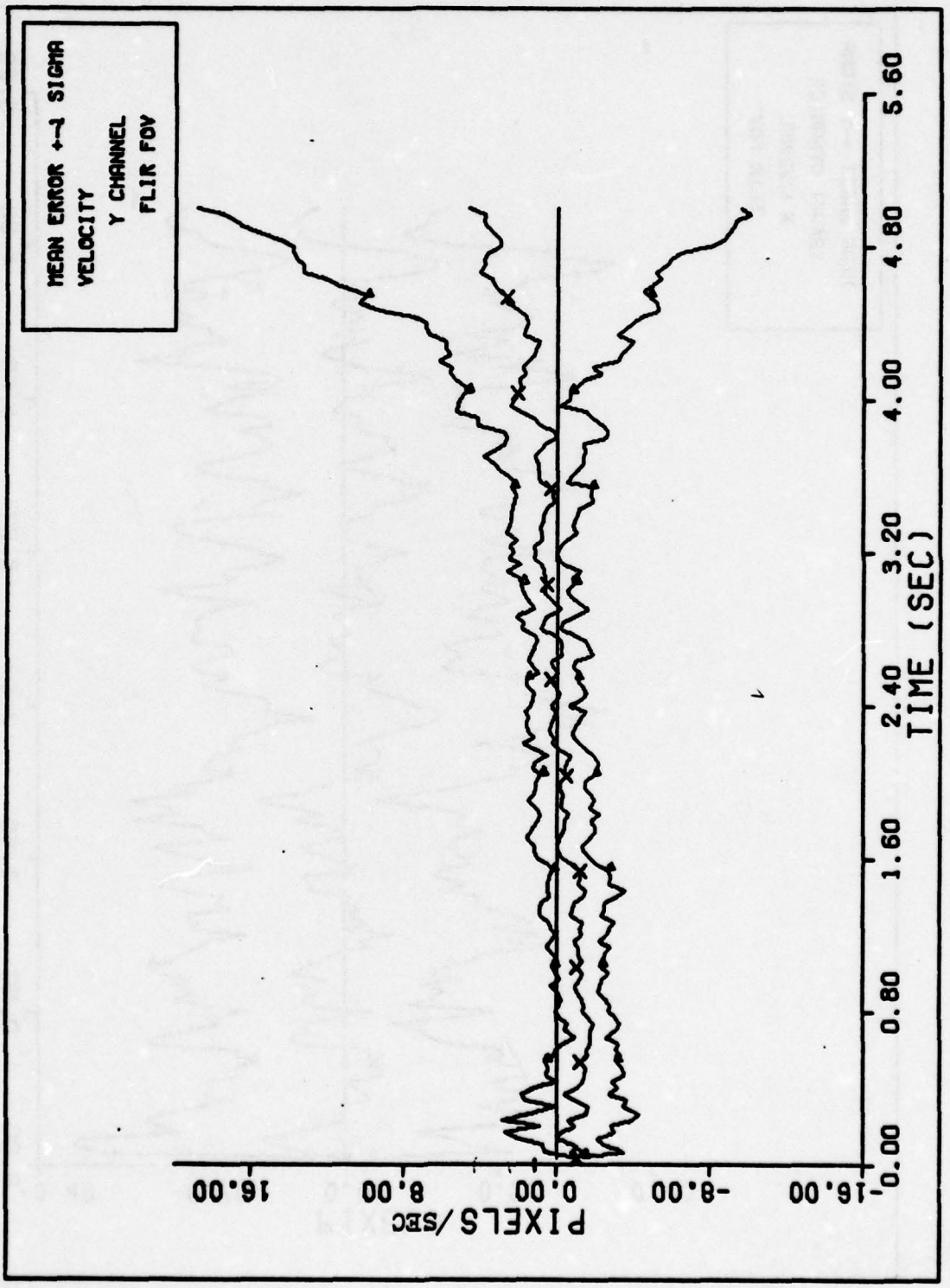
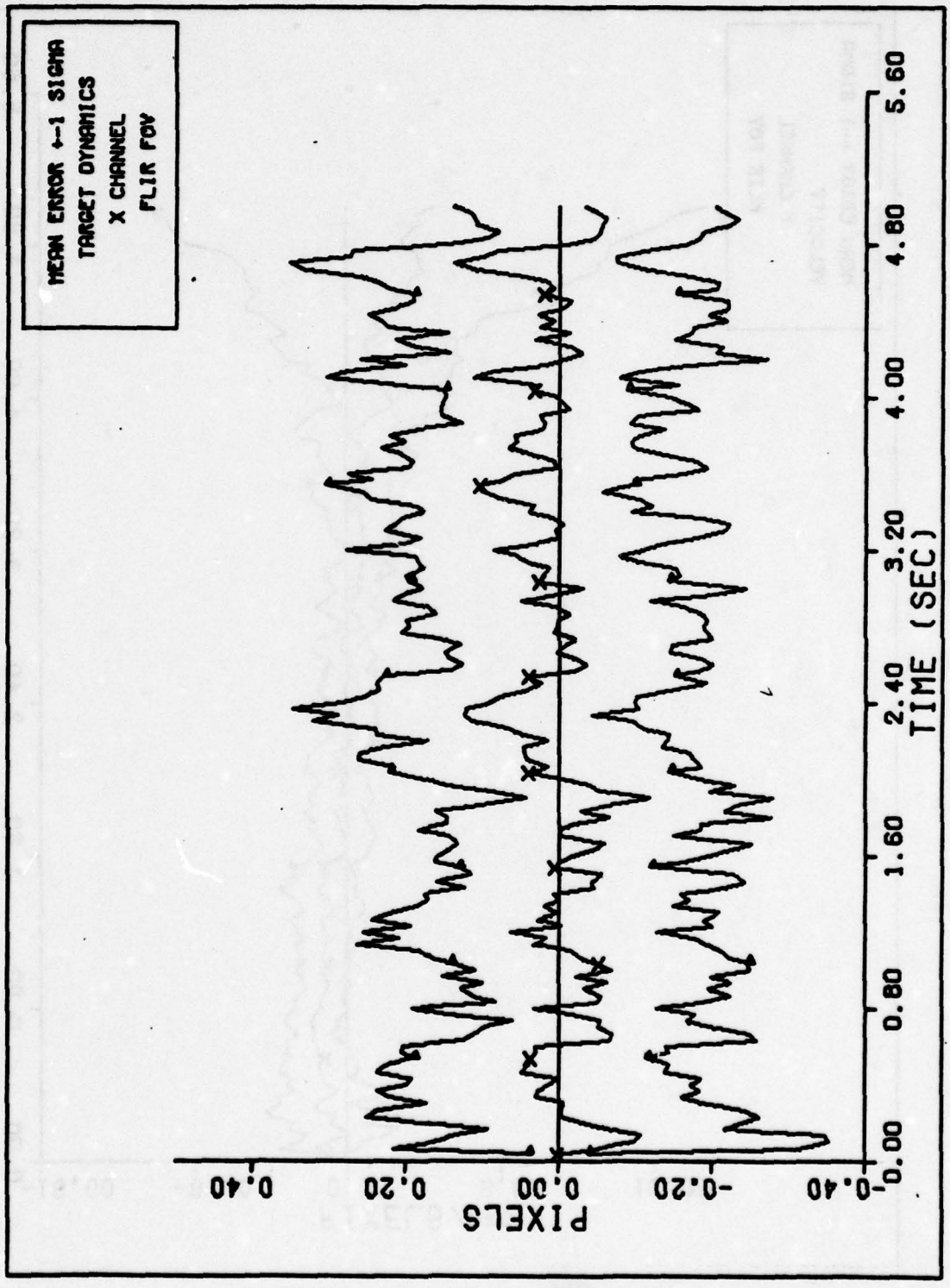
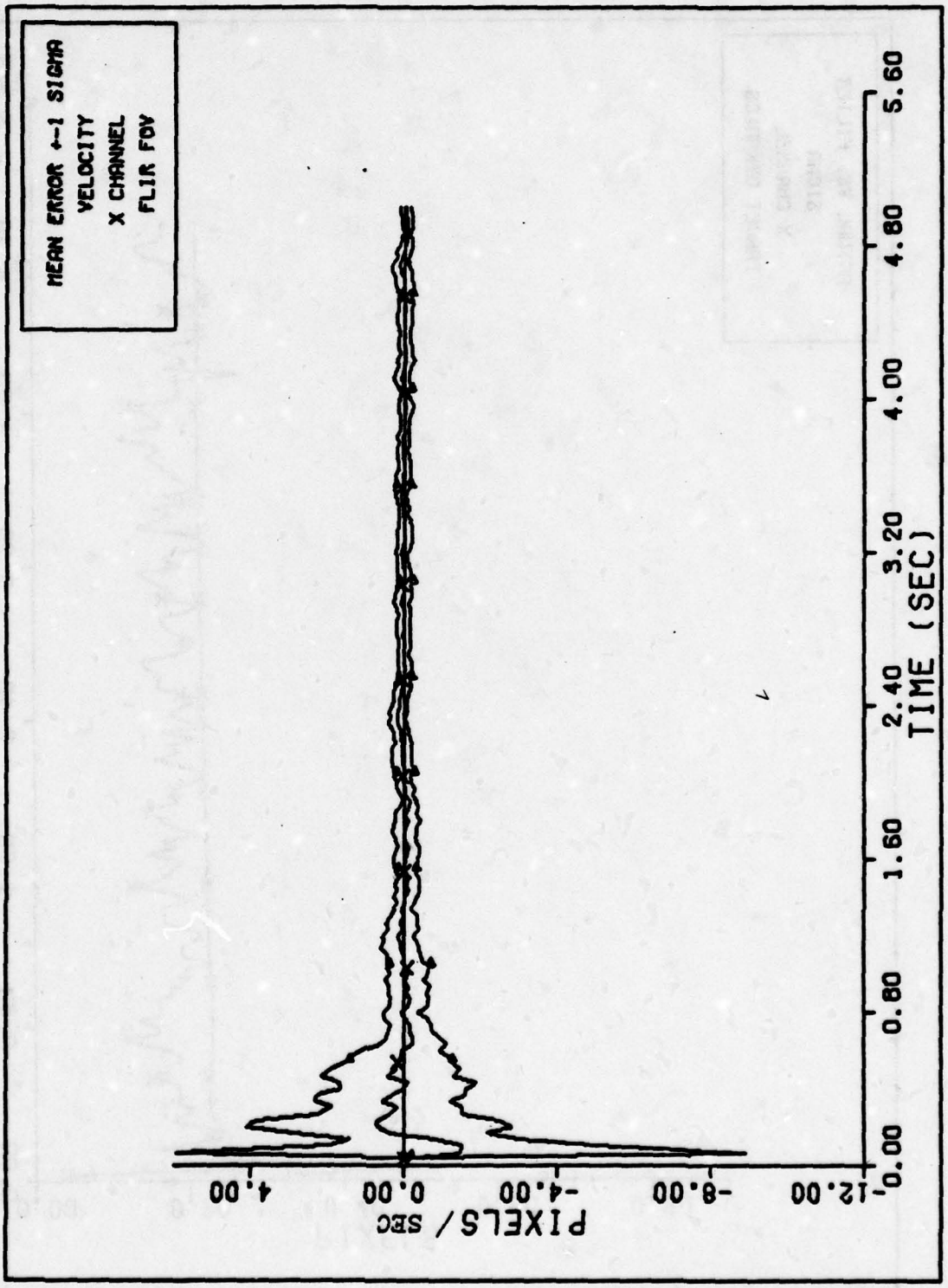


Figure 31f. Case 33 Performance Plot



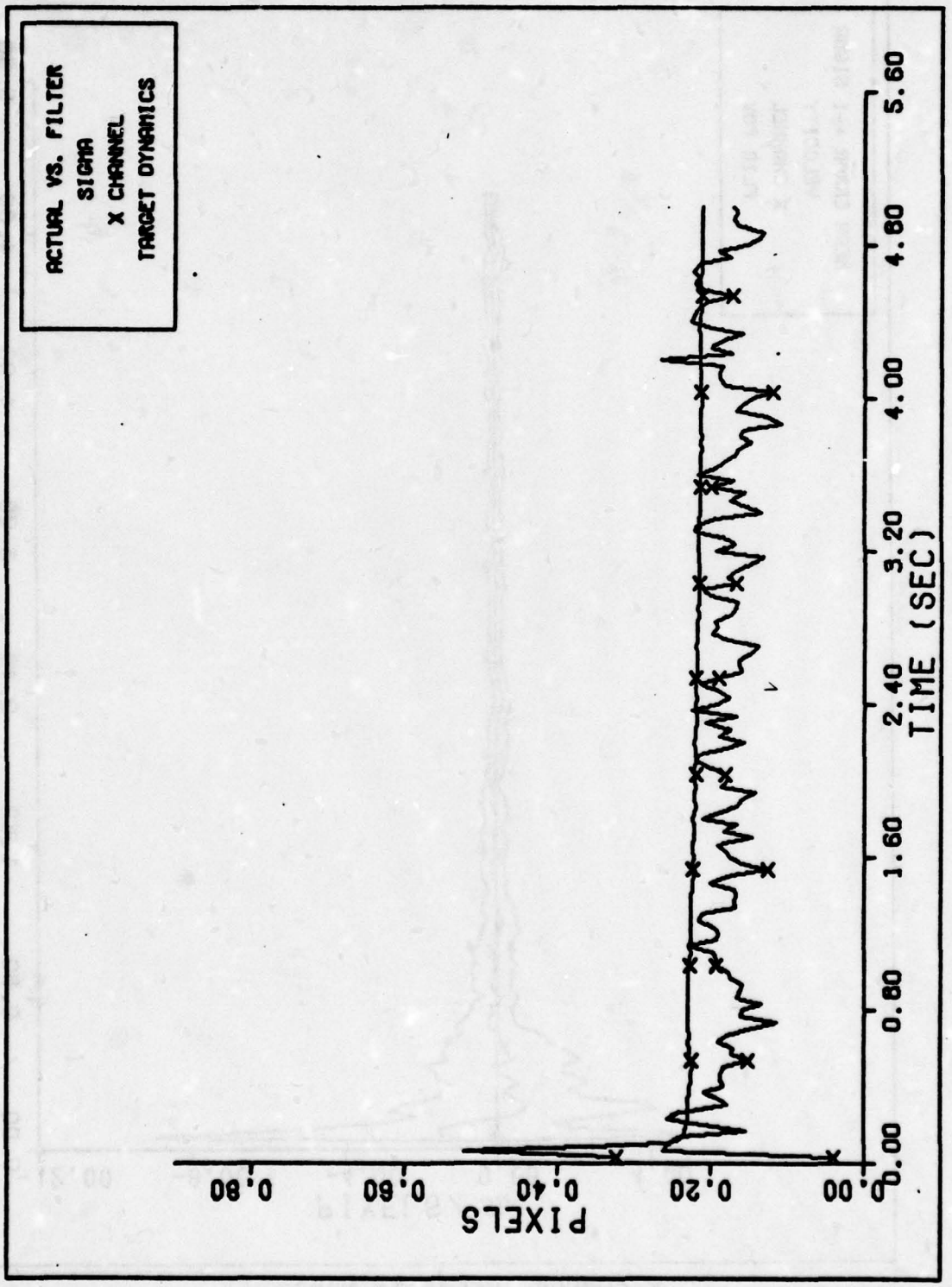
X CHANNEL DYNAMICS ERROR (S/N=12.5)

Figure 32a. Case 34 Performance Plot



Y CHANNEL VELOCITY ERROR

Figure 32b. Case 34 Performance Plot



FILTER VS. ACTUAL SIGMA PLOT

Figure 32c. Case 34 Performance Plot

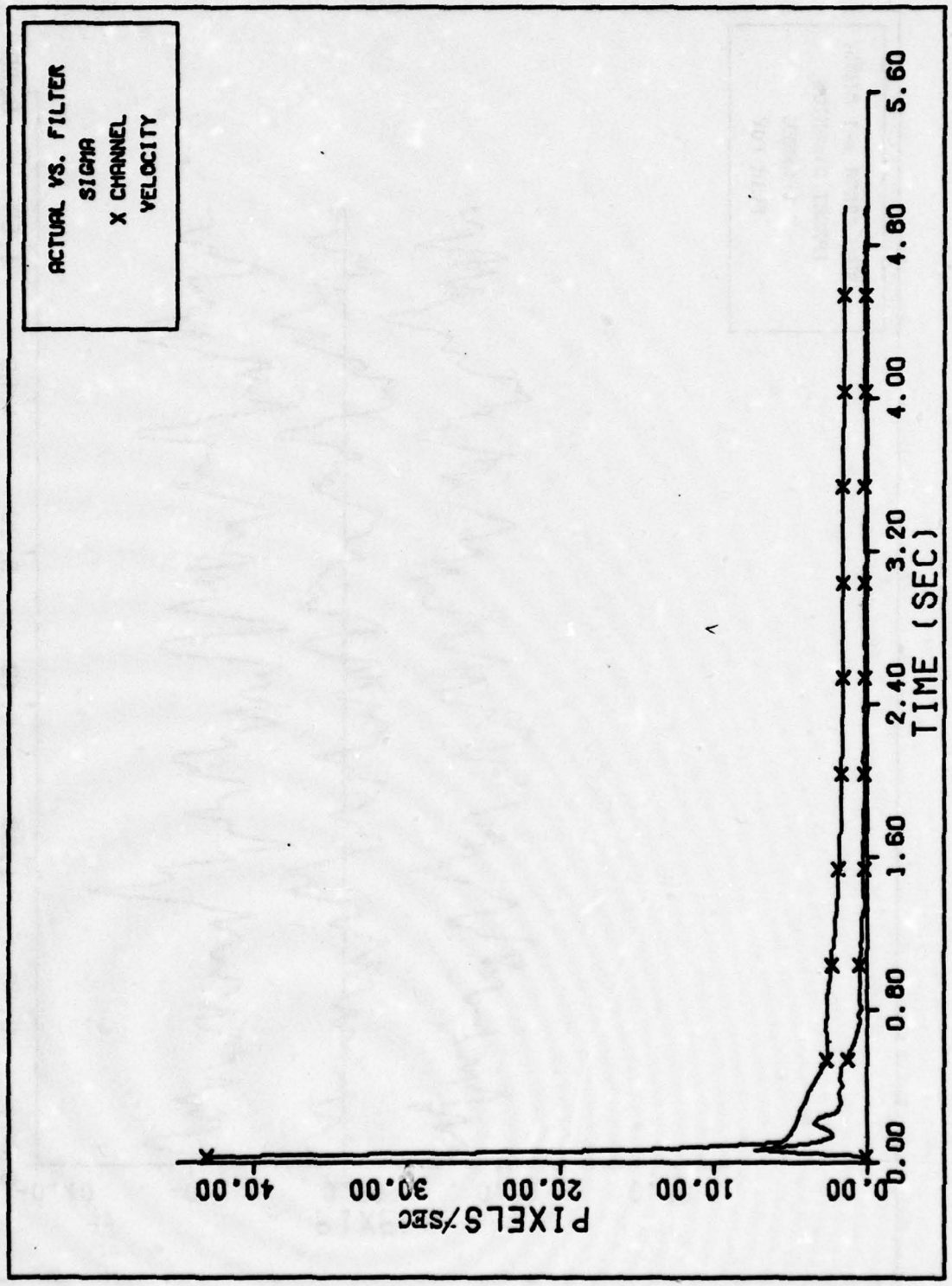
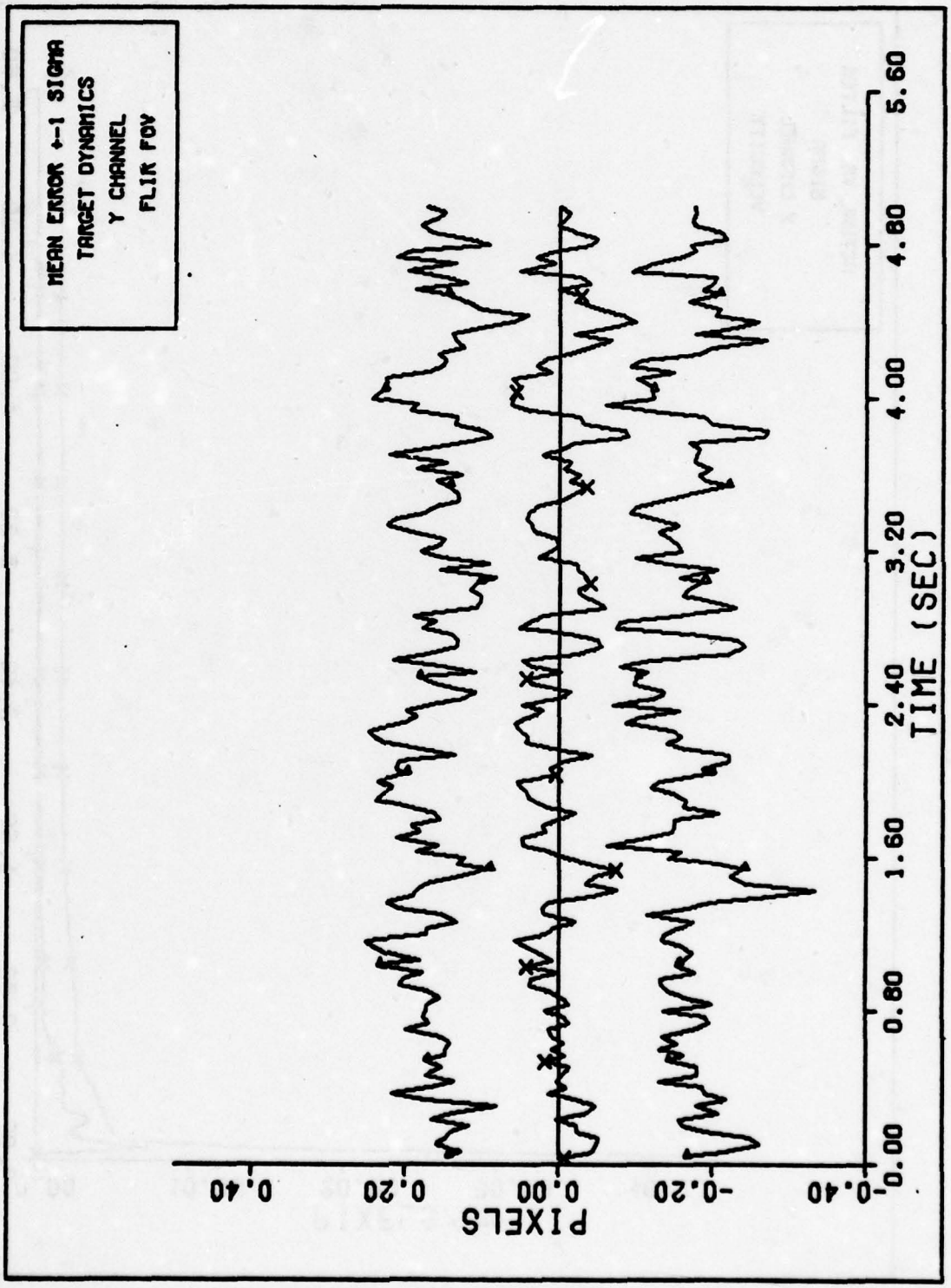


Figure 32d. Case 34 Performance Plot



Y CHANNEL DYNAMICS ERROR

Figure 32e. Case 34 Performance Plot

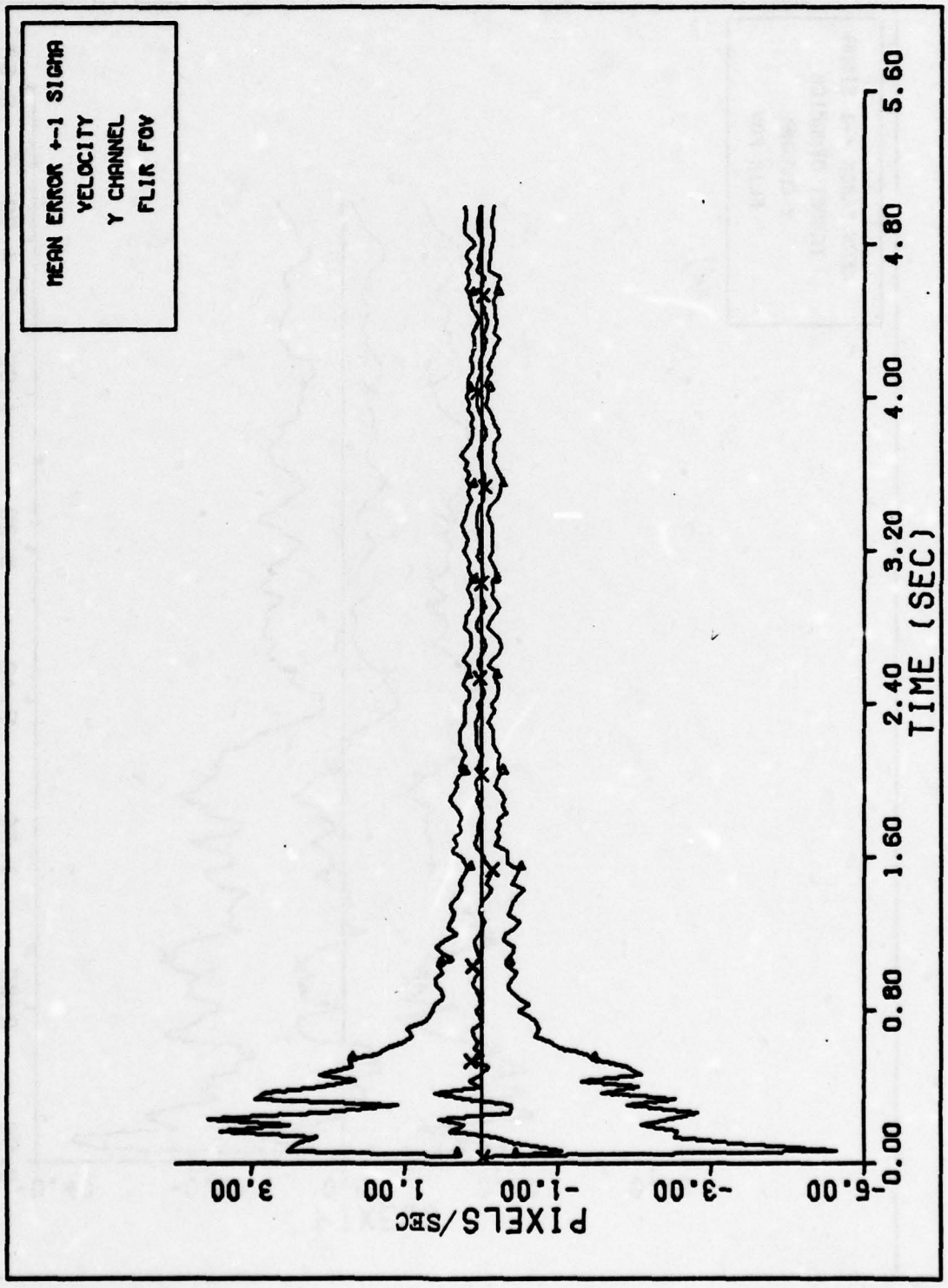
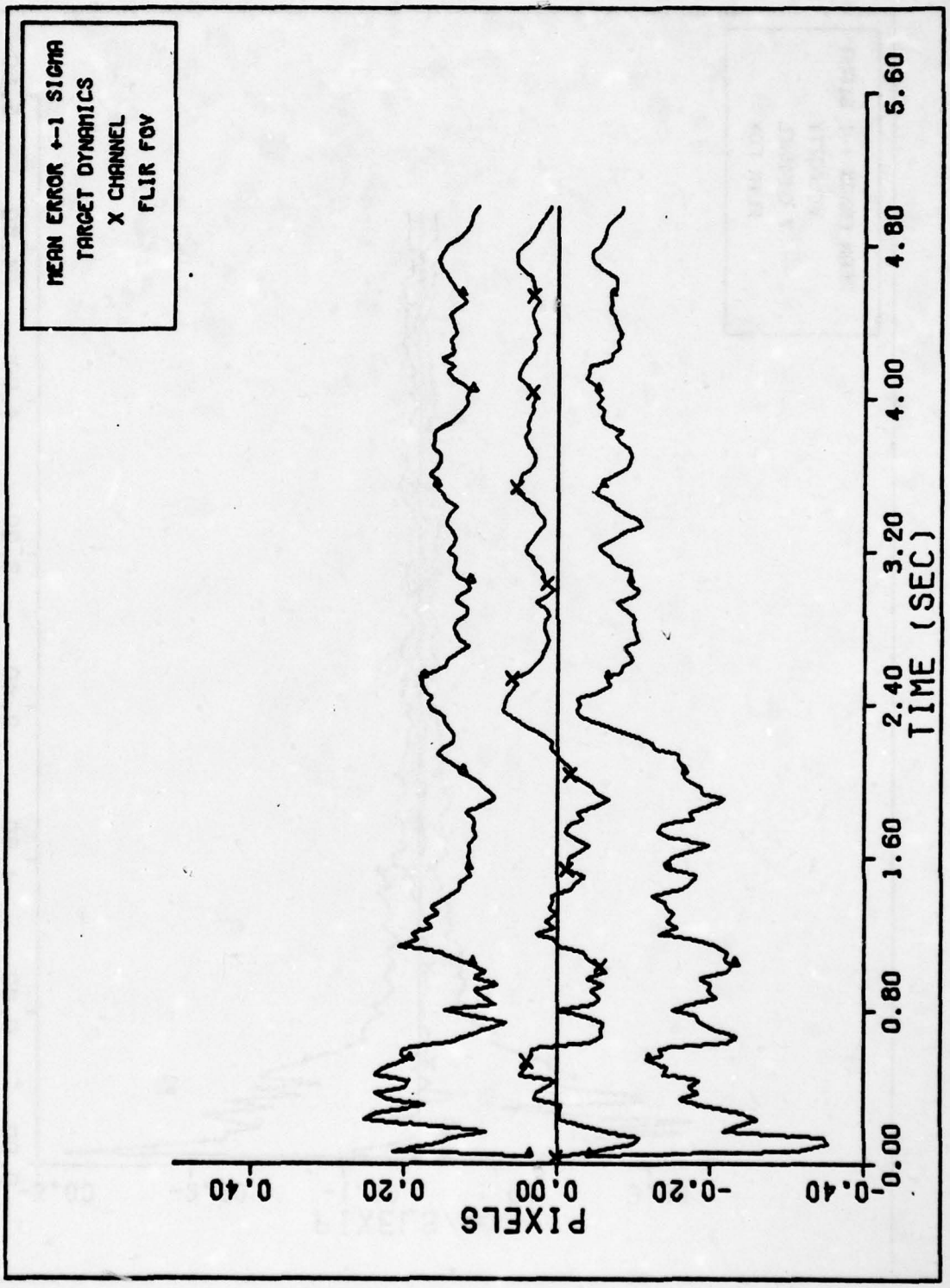
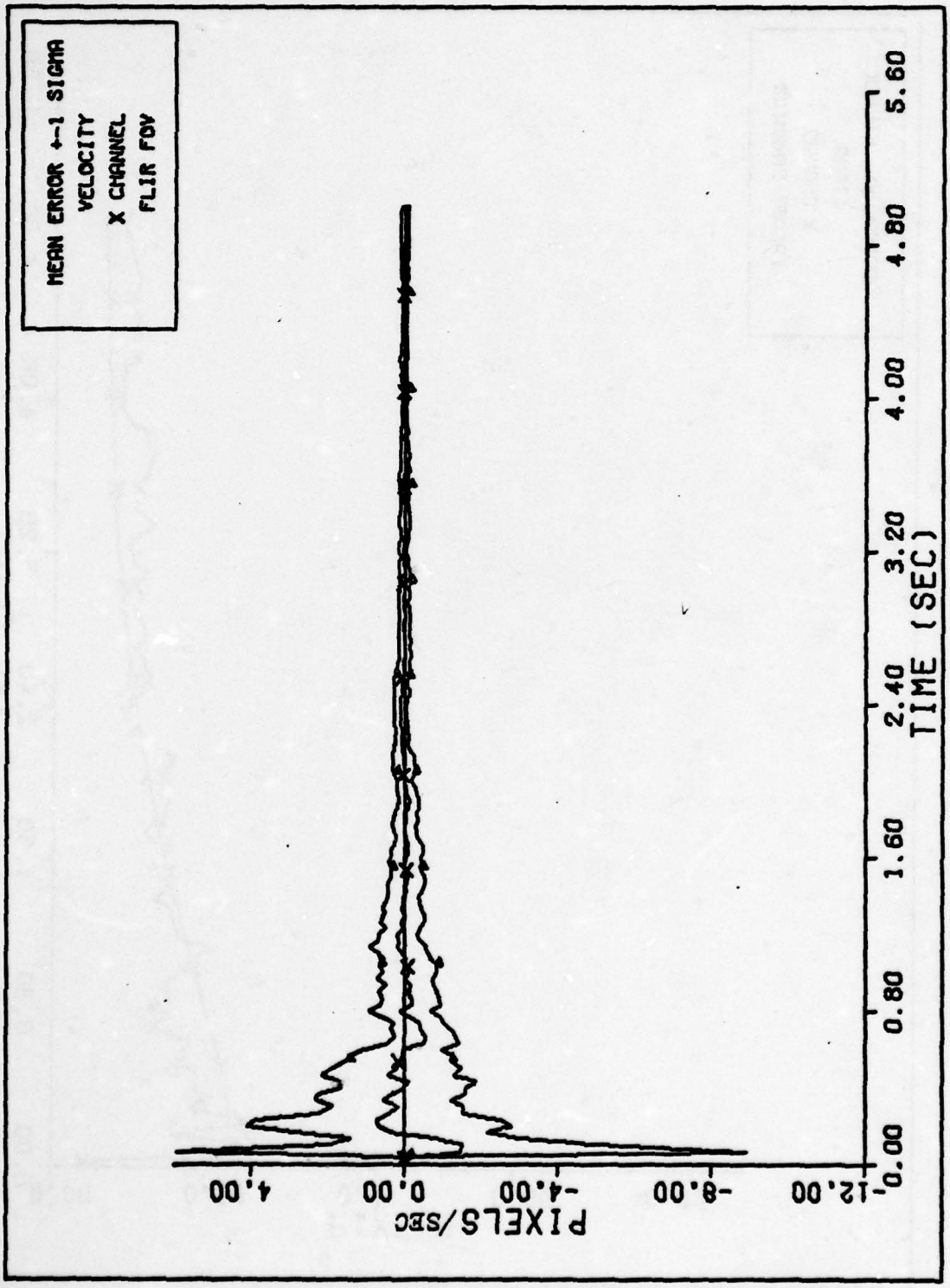


Figure 32f. Case 34 Performance Plot



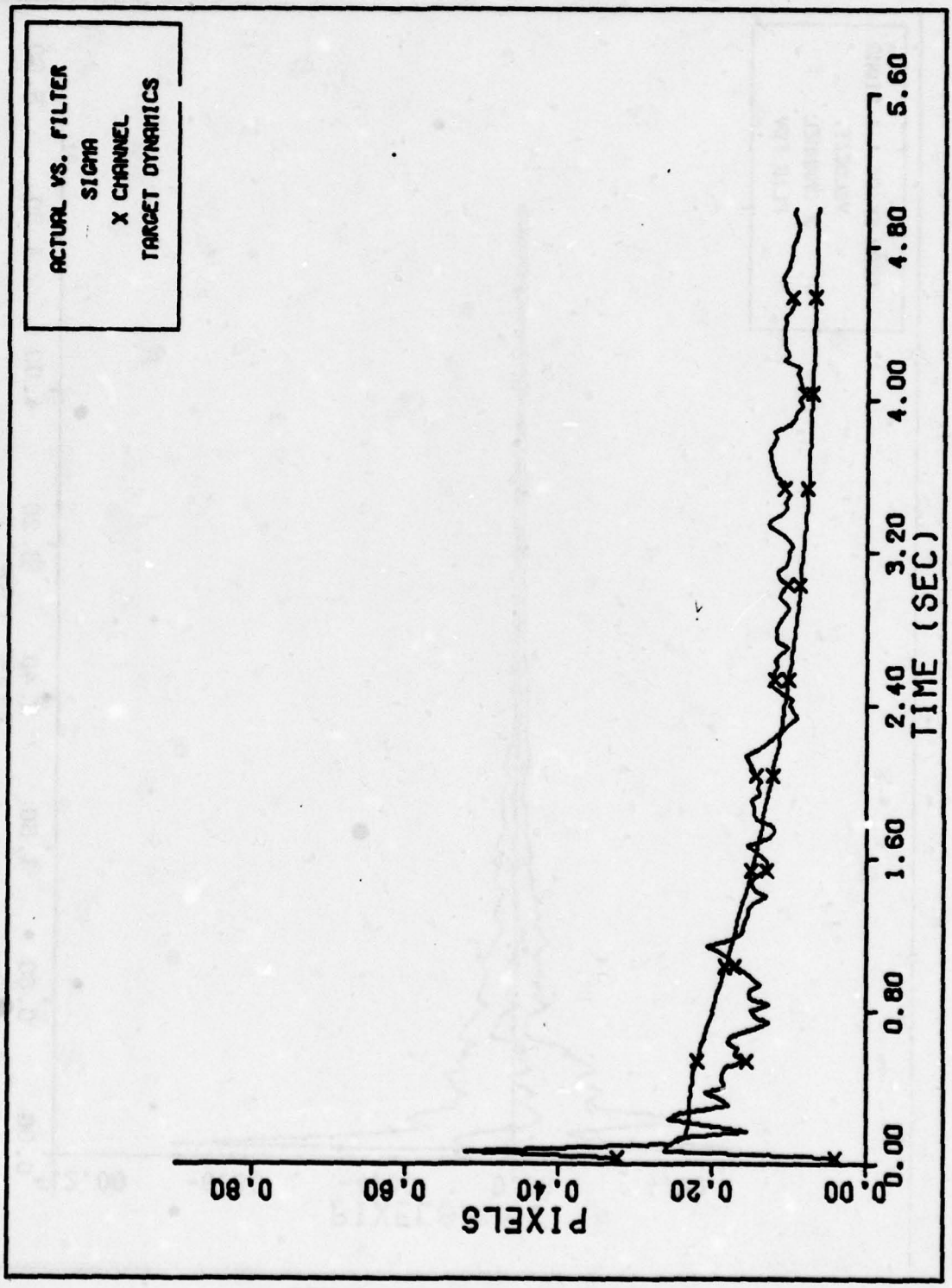
X CHANNEL DYNAMICS ERROR (S/N=12.5)

Figure 33a. Case 35 Performance Plot



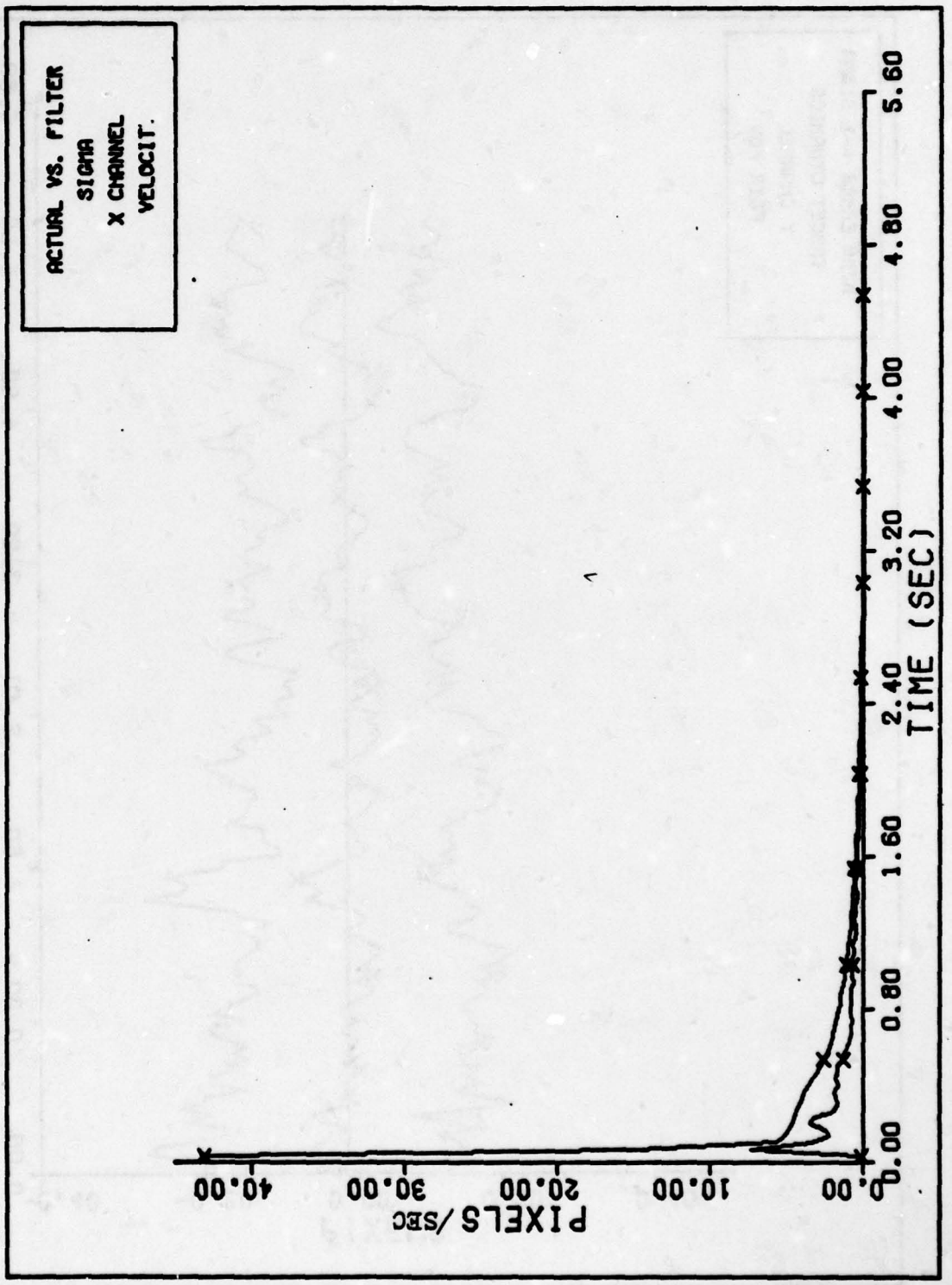
Y CHANNEL VELOCITY ERROR

Figure 33b. Case 35 Performance Plot



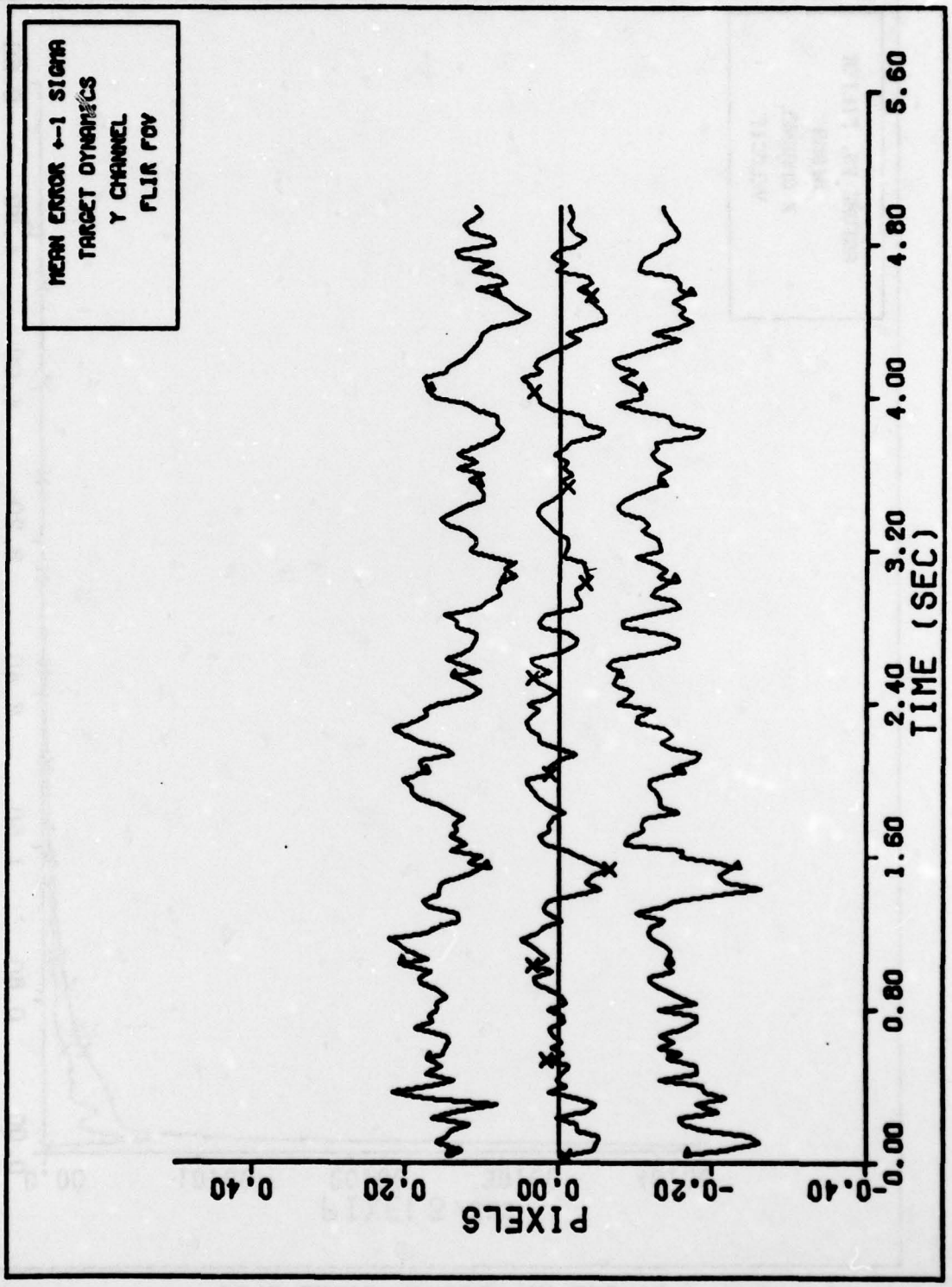
FILTER VS. ACTUAL SIGMA PLOT

Figure 33c. Case 35 Performance Plot



FILTER VS. ACTUAL SIGMA PLOT

Figure 33d. Case 35 Performance Plot



Y CHANNEL DYNAMICS ERROR

Figure 33e. Case 35 Performance Plot

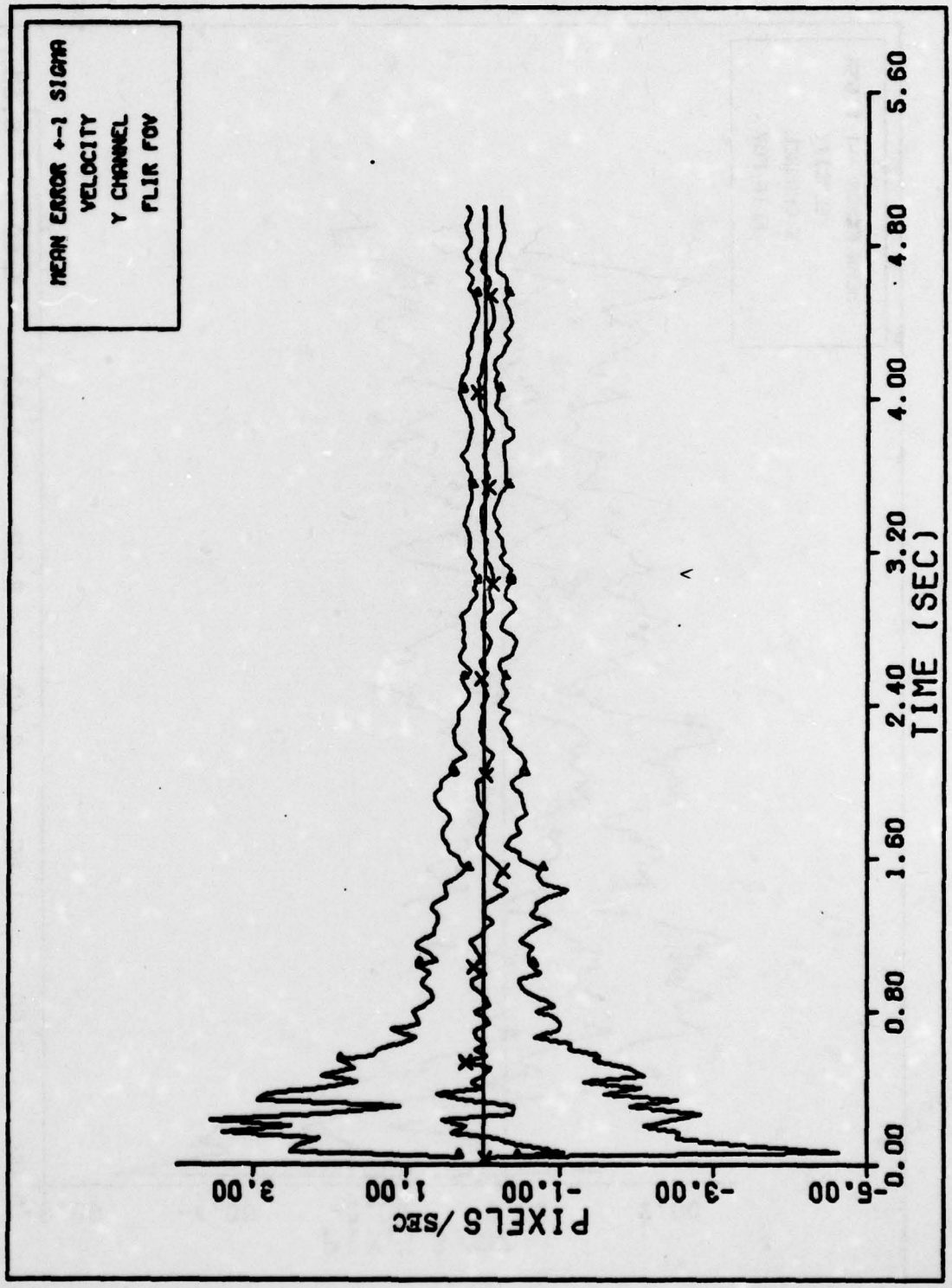
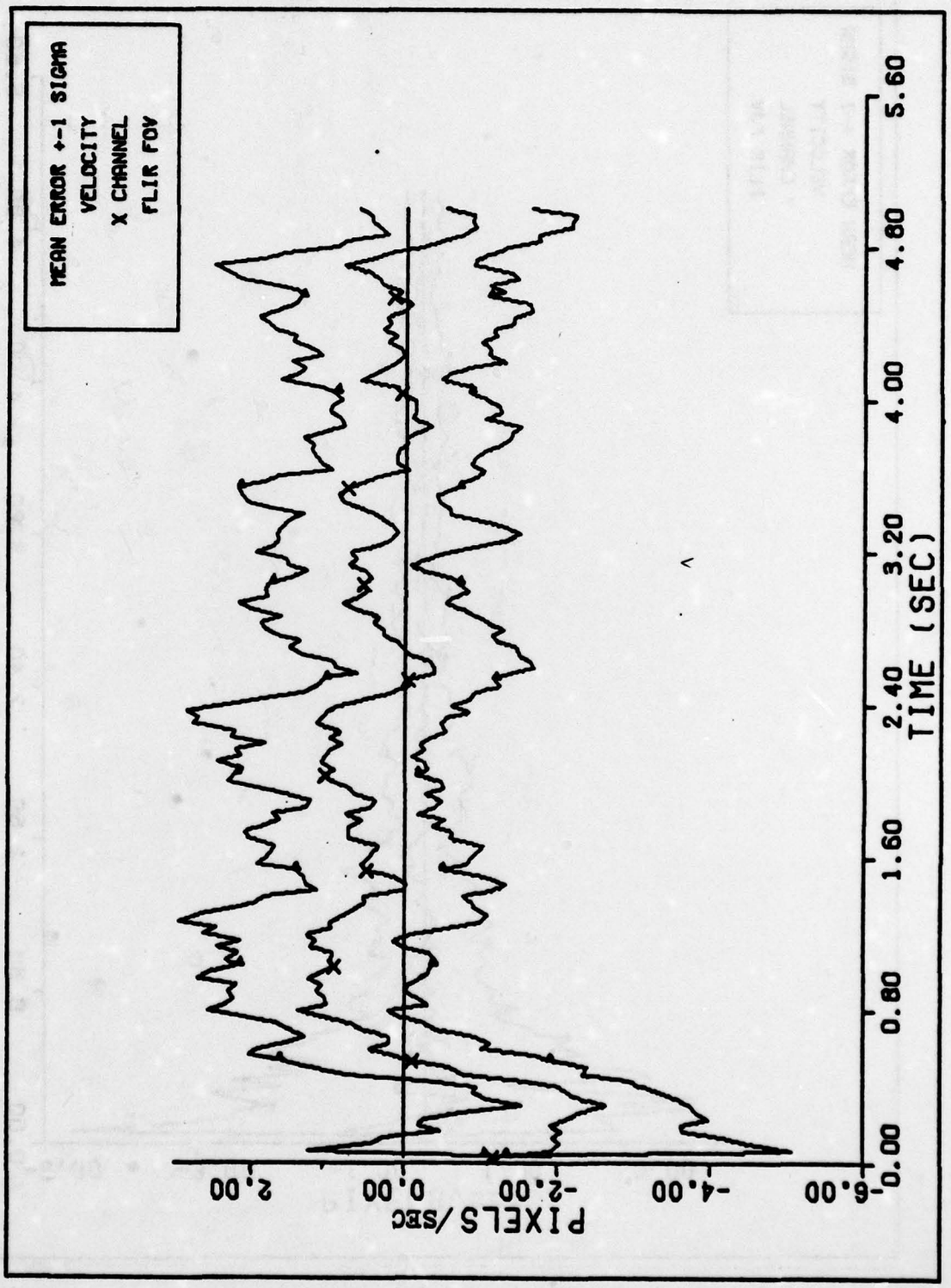
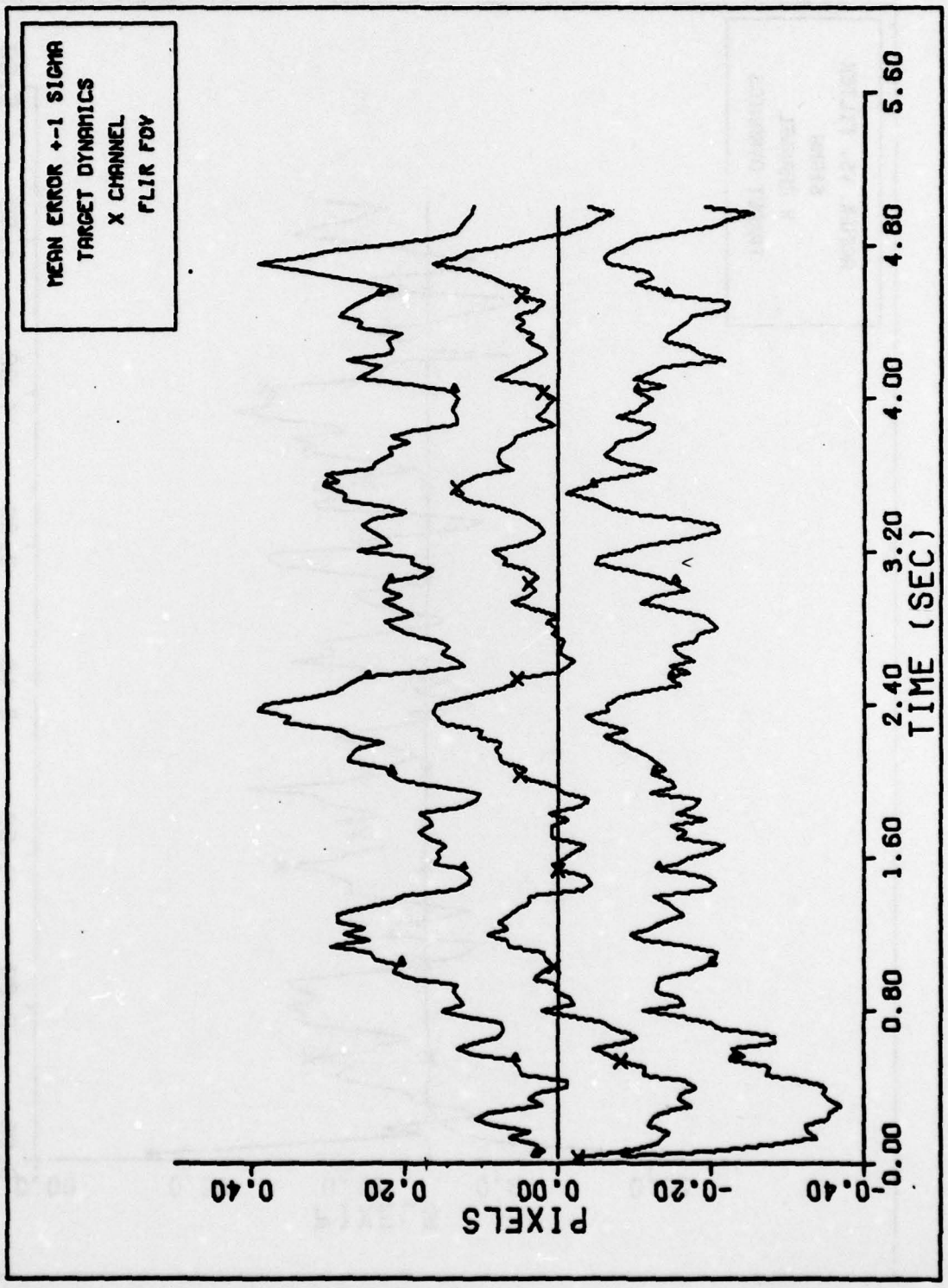


Figure 33f. Case 35 Performance Plot



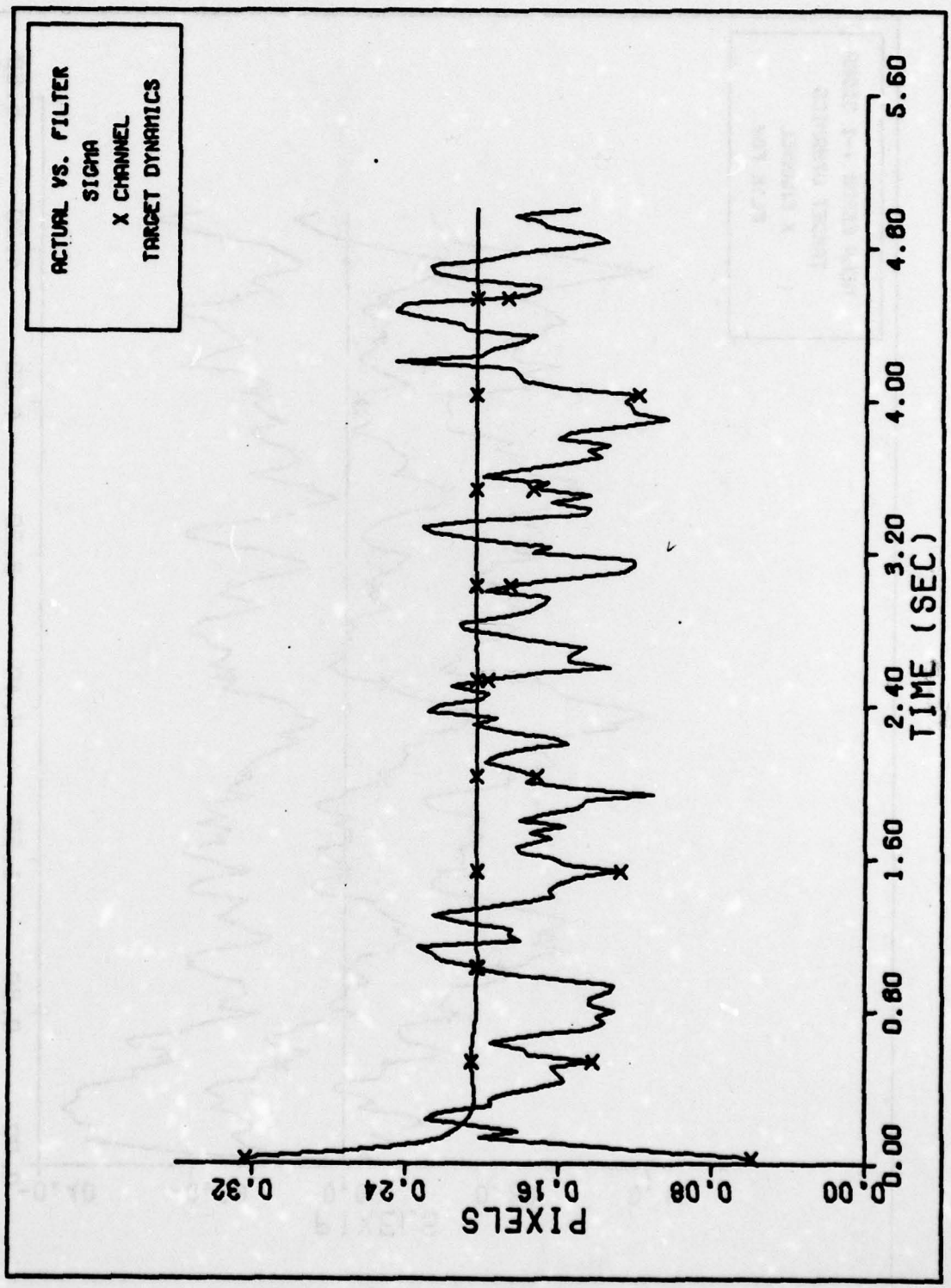
X CHANNEL VELOCITY ERROR (S/N=12.5)

Figure 34a. Case 36 Performance Plot



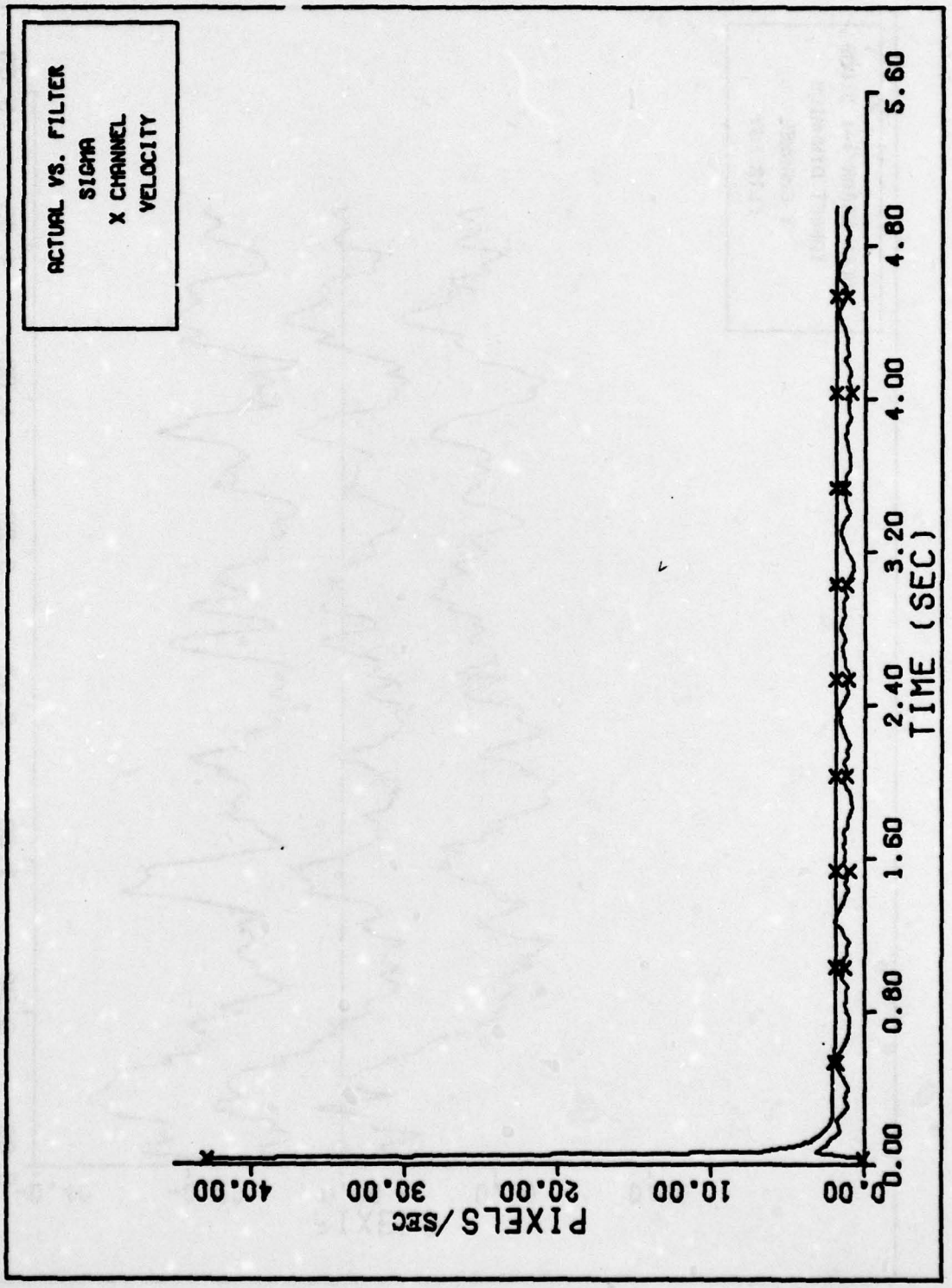
X CHANNEL DYNAMICS ERROR

Figure 34b. Case 36 Performance Plot



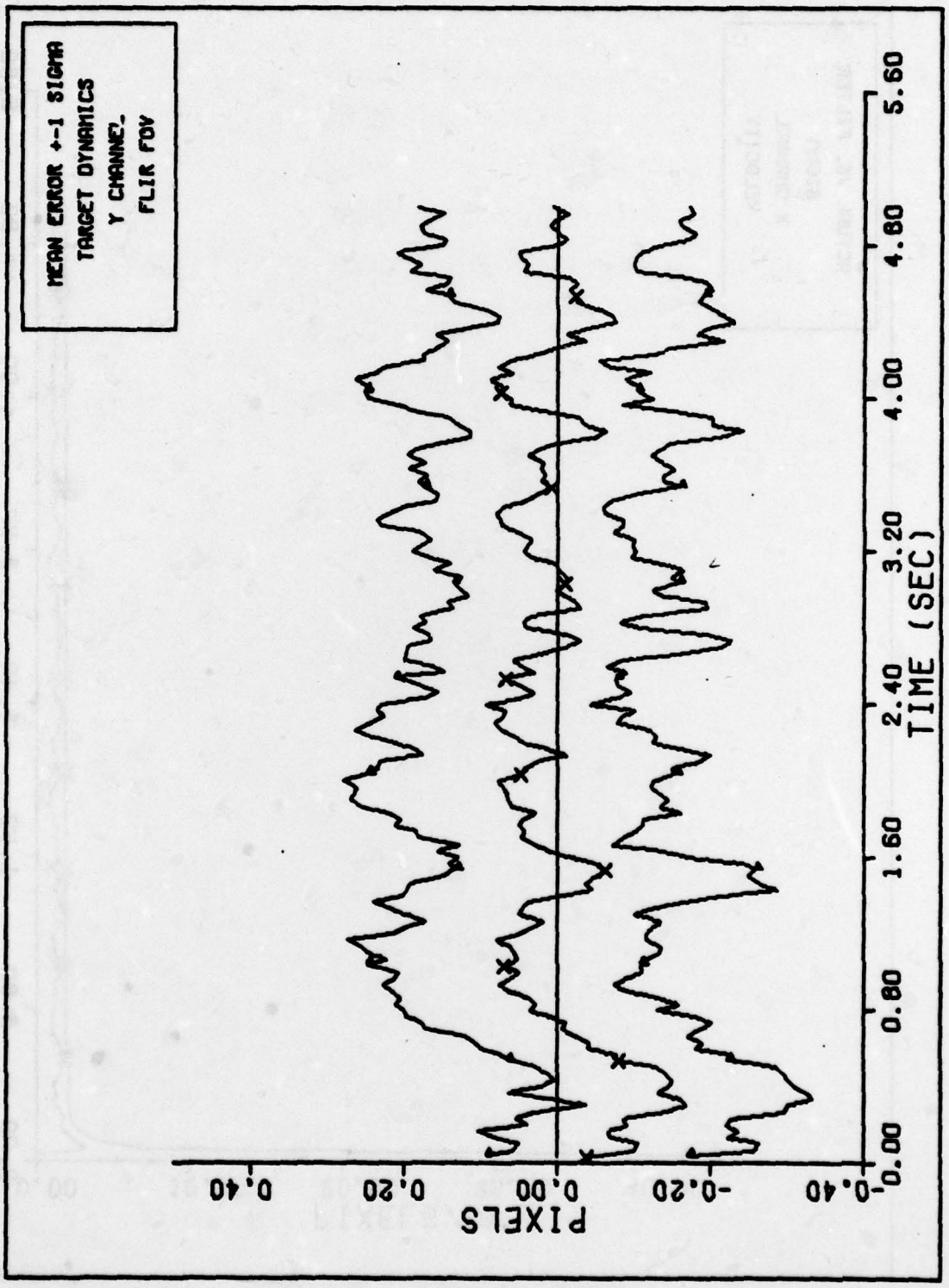
FILTER VS. ACTUAL SIGMA PLOT

Figure 34c. Case 36 Performance Plot



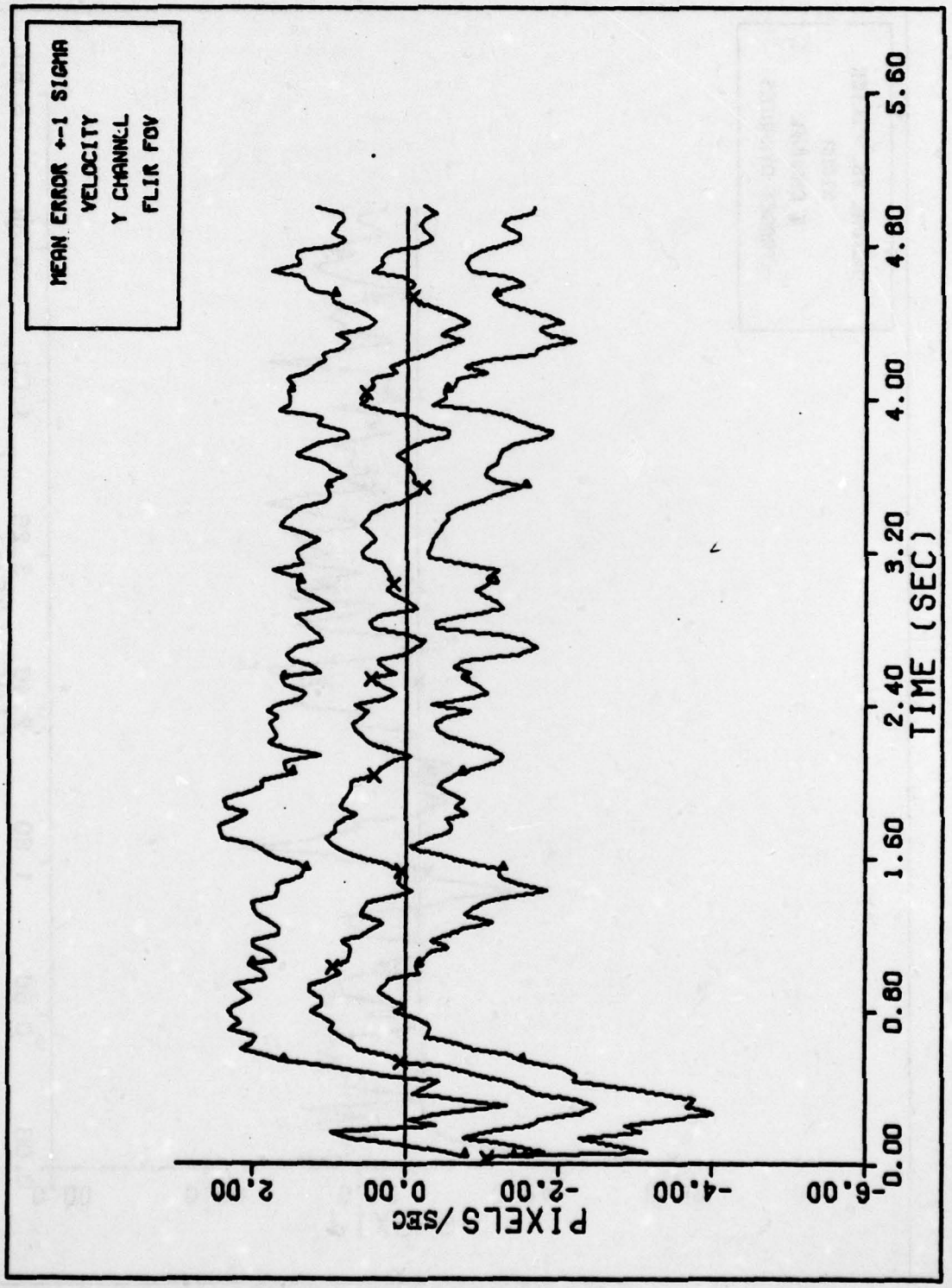
FILTER VS. ACTUAL SIGMA PLOT

Figure 34d. Case 36 Performance Plot



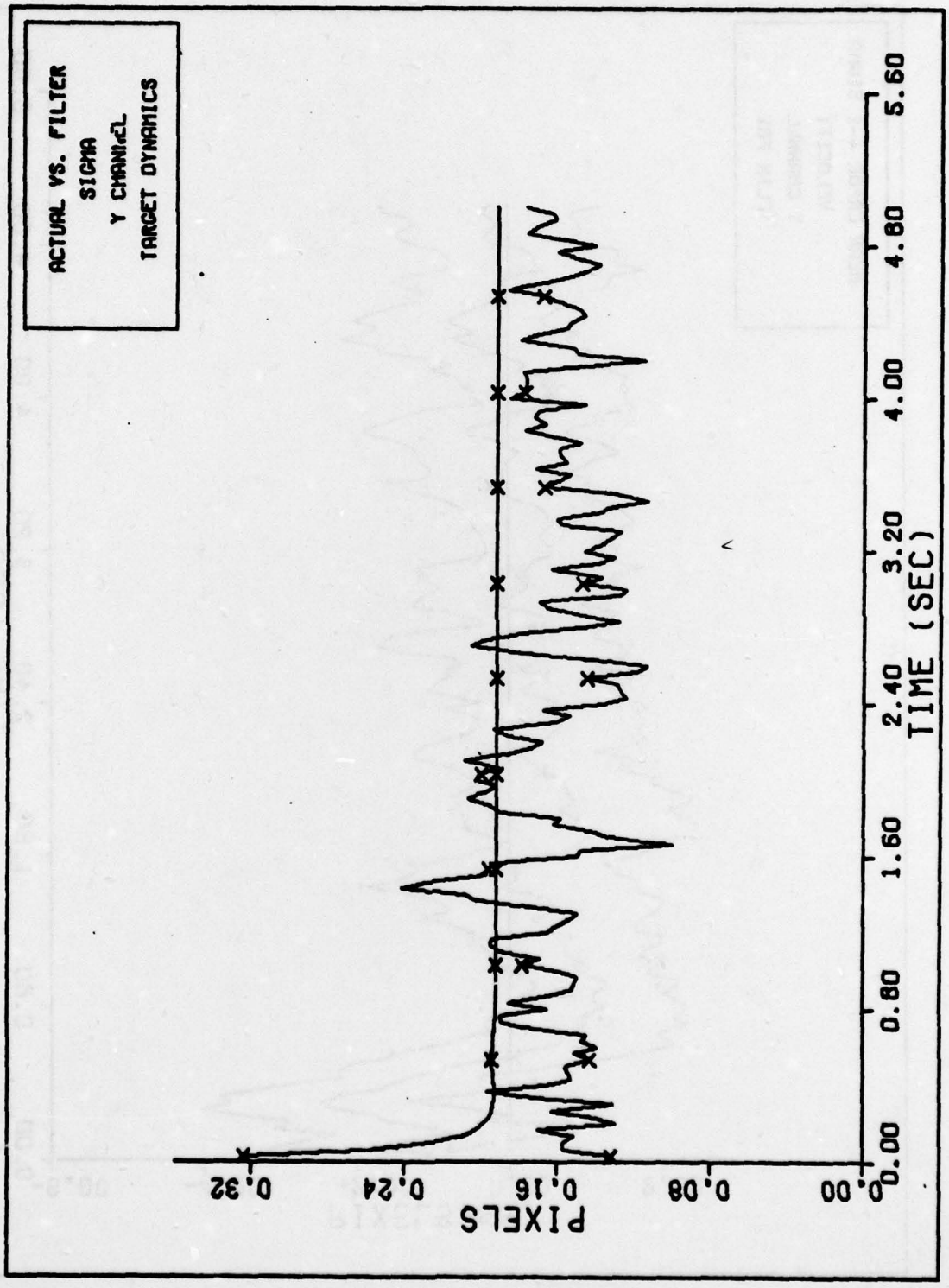
Y CHANNEL DYNAMICS ERROR

Figure 34e. Case 36 Performance Plot



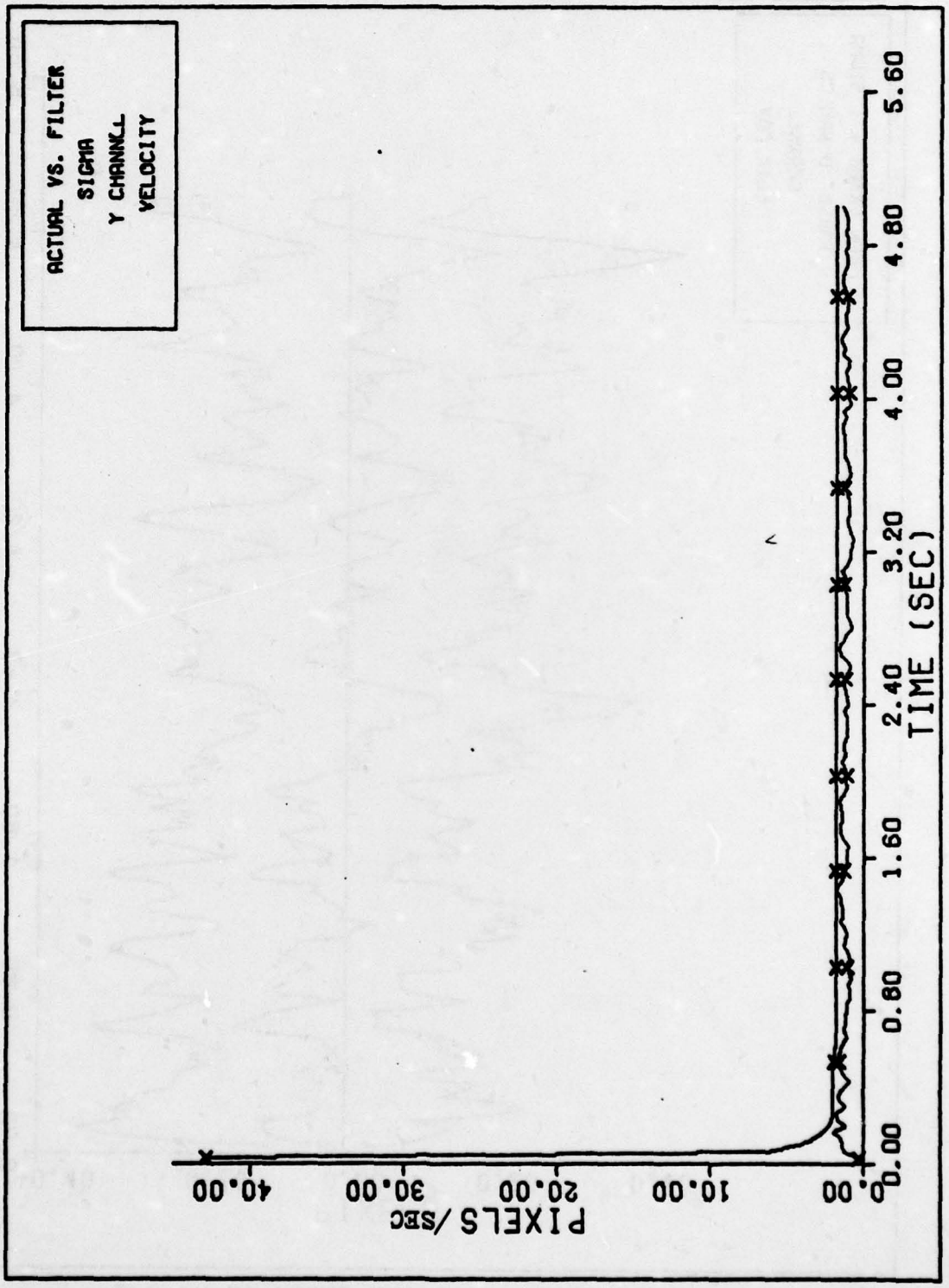
Y CHANNEL VELOCITY ERROR

Figure 34f. Case 36 Performance Plot



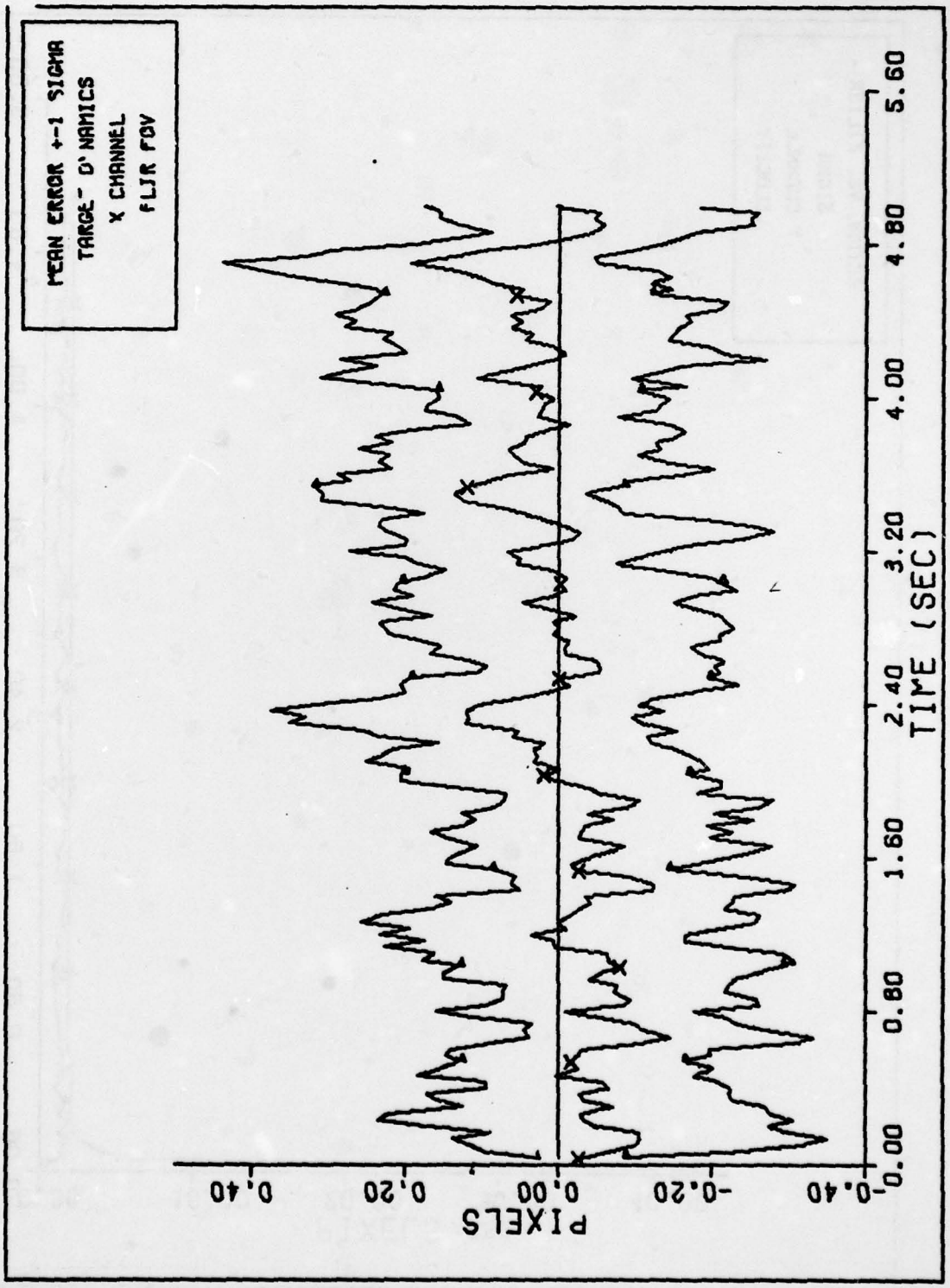
FILTER VS. ACTUAL SIGMA PLOT

Figure 34g. Case 36 Performance Plot



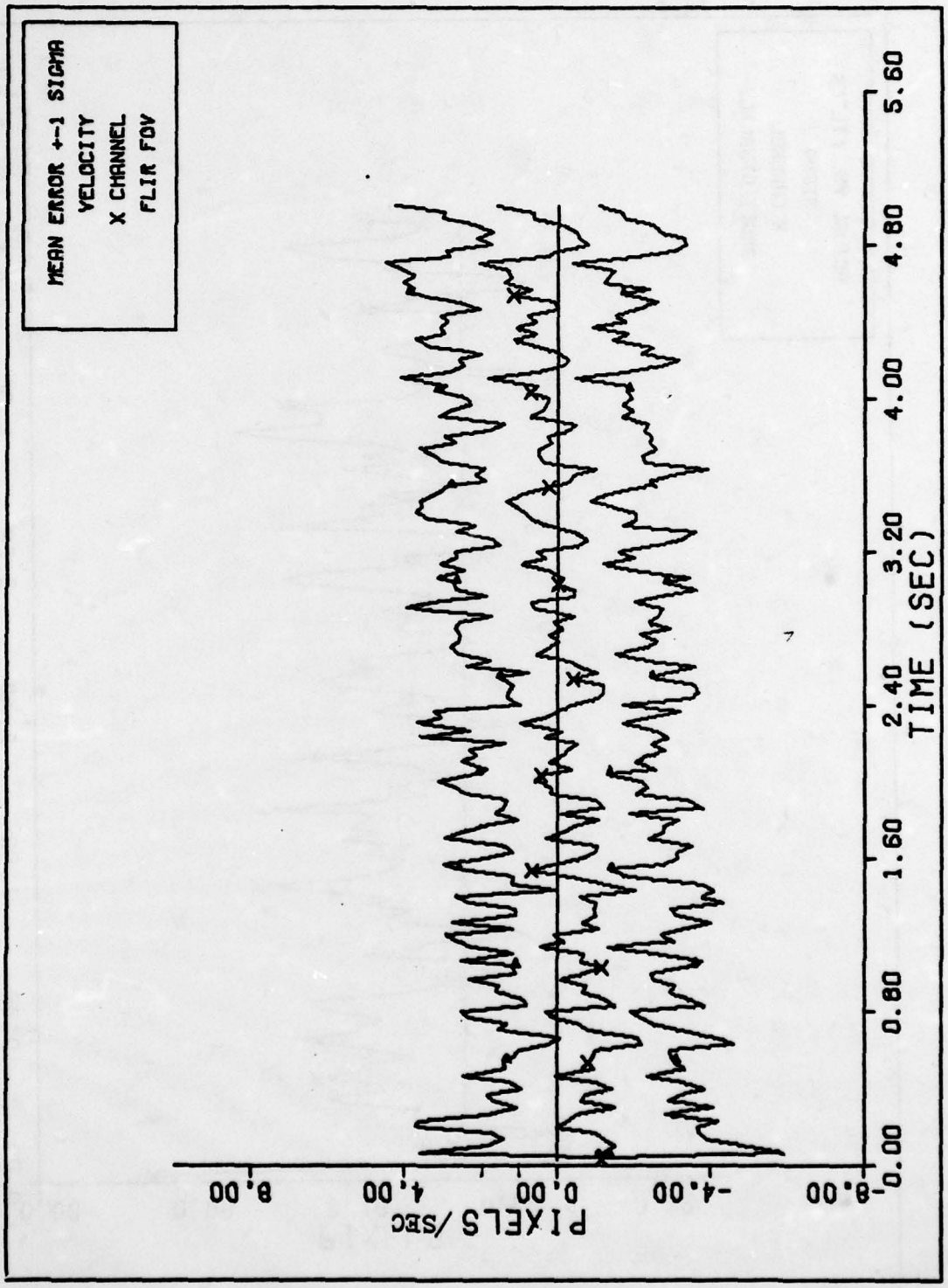
FILTER VS. ACTUAL SIGMA PLOT

Figure 34h. Case 36 Performance Plot



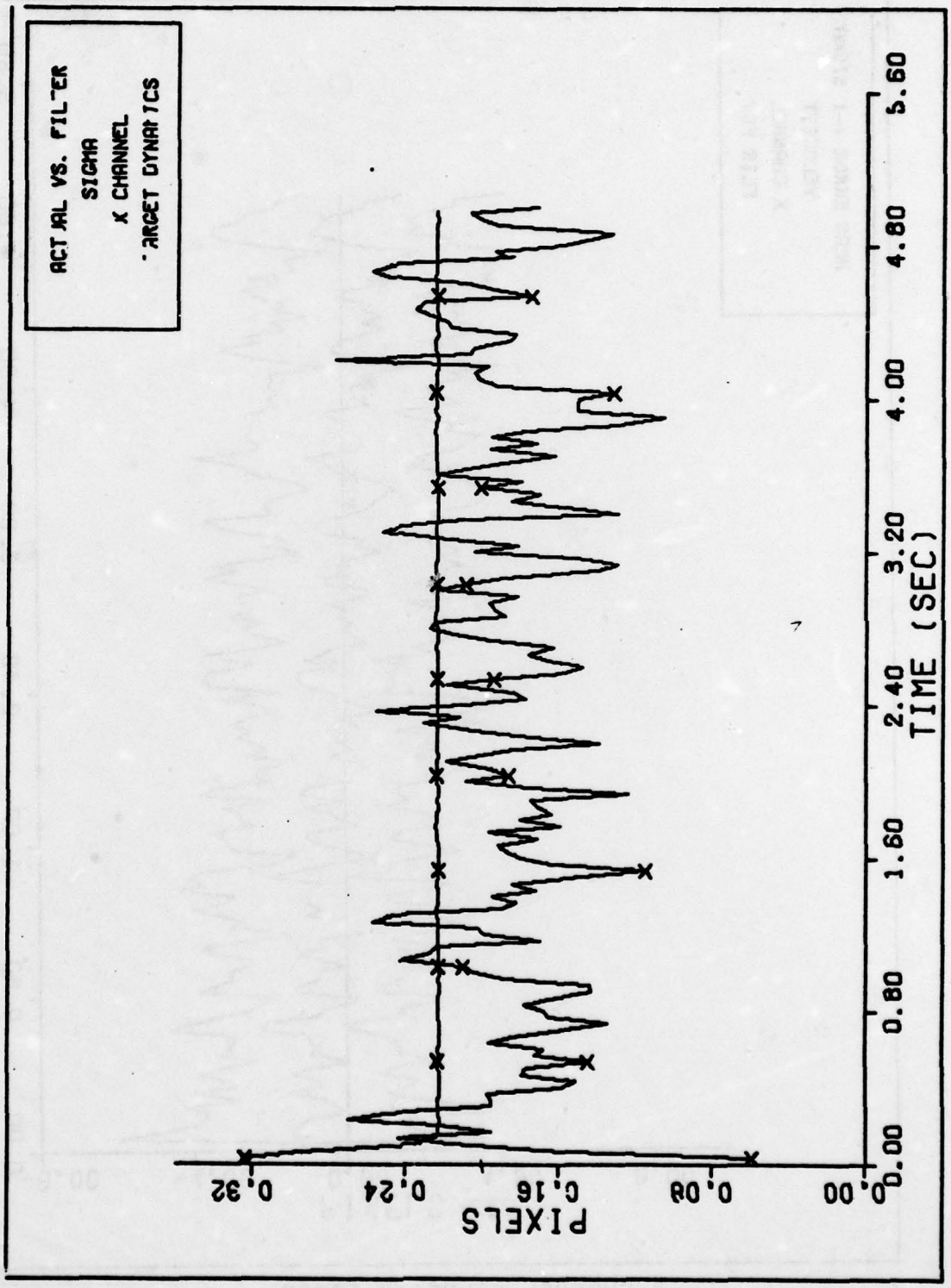
X CHANNEL DYNAMICS ERROR (S/N=12.5)

Figure 35a. Case 37 Performance Plot



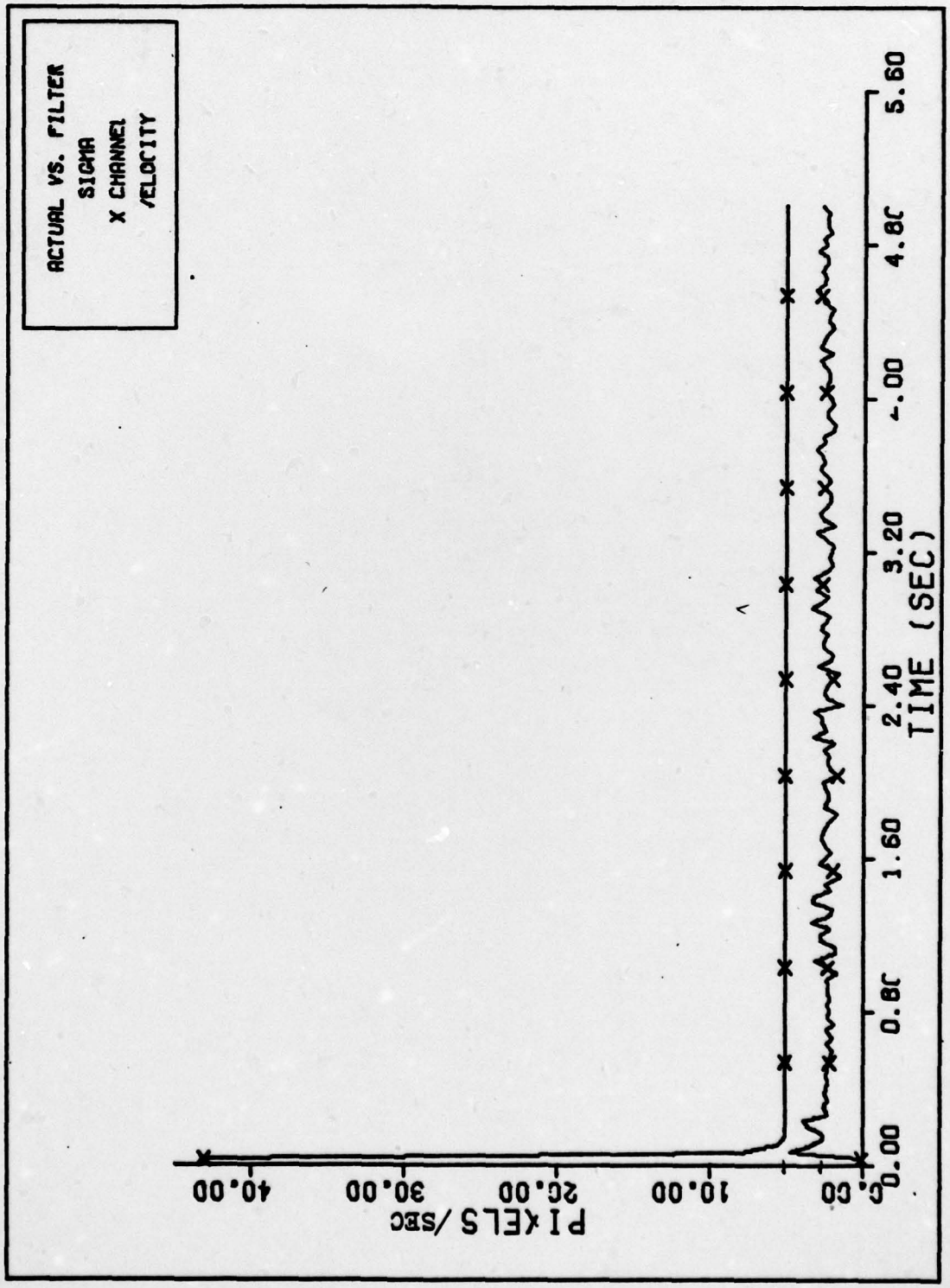
X CHANNEL VELOCITY ERROR

Figure 35b. Case 37 Performance Plot



FILTER VS. ACTUAL SIGMA PLOT

Figure 35c. Case 37 Performance Plot



FILTER VS. ACTUAL SIGMA PLOT

Figure 35d. Case 37 Performance Plot